

UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA



TRABAJO DE GRADUACIÓN:

**“IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DE
INFORMACIÓN EN EL CEMENTERIO SANTA ISABEL”**

PRESENTADO POR:

CHÁVEZ OLIVARES, LUIS ANTONIO
CLAVEL TOLEDO, LUIS ANTONIO
VÁSQUEZ CASTANEDA, GUILLERMO RAFAEL

PARA OPTAR AL GRADO DE:

INGENIERO DE SISTEMAS INFORMÁTICOS

DOCENTE DIRECTOR:

ING. CARLOS STANLEY LINARES PAULA

SANTA ANA, MAYO DE 2011

RECTOR

ING. Y MSC. RUFINO QUEZADA SÁNCHEZ

VICE-RECTOR ACADÉMICO

ARQ. Y MASTER MIGUEL ANGEL PÉREZ RAMOS

VICE-RECTOR ADMINISTRATIVO

LICDO. Y MASTER OSCAR NOÉ NAVARRETE

SECRETARIO GENERAL

LICDO. DOUGLAS VLADIMIR ALFARO CHAVEZ

FISCAL GENERAL

DR. RENE MADECADEL PERLA JIMÉNEZ

DECANO

LICDO. JORGE MAURICIO RIVERA

VICE-DECANO

LICDO. Y MASTER ELADIO EFRAIN ZACARIAS ORTEZ

SECRETARIO DE FACULTAD

LICDO. VICTOR HUGO MERINO QUEZADA

JEFE DEL DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA

ING. RAUL ERNESTO MARTINEZ BERMUDEZ

TRABAJO DE GRADO APROBADO POR:

DOCENTE DIRECTOR:

ING. CARLOS STANLEY LINARES PAULA

AGRADECIMIENTOS GENERALES

Al personal del Cementerio Santa Isabel por darnos la oportunidad, la confianza y las herramientas necesarias para la realización de este proyecto.

A nuestro asesor Ingeniero Carlos Stanley Paula, gracias por darnos una guía en todos los aspectos para la buena realización del presente trabajo de grado.

Al Ingeniero Raúl Ernesto Martínez Bermúdez jefe del departamento de Ingeniería y Arquitectura de la Facultad Multidisciplinaria de Occidente, por todo su apoyo en el proceso del trabajo de grado.

AGRADECIMIENTOS

A **Dios** por regalarme la oportunidad de tener la vida que me dió, por ser mi socorro en todo momento y permitirme sobreponerme a las adversidades que la vida y la Universidad me presentaron. Por permitirme lograr mis objetivos académicos sin claudicar a pesar que muchas veces viví situaciones difíciles.

A mi madre **Rina del Carmen Vda. De Chávez**, por ser padre y madre desde que mi padre murió, por sacrificarse y esforzarse cada día trabajando para regalarme la oportunidad de estudiar y formarme como Ingeniero. Por escucharme cuando lo necesité y apoyarme en todos los proyectos que he emprendido a lo largo de mi vida, dándome siempre todo lo que estuvo en sus manos para hacerme feliz.

A mi padre **Luis Antonio Chávez Turcios (QDDG)** por cuidarme en todo momento. Por inculcarme que el éxito solo se logra a través del trabajo constante y el empeño que ponemos al realizar nuestras actividades.

A mi hermana **Ana Jamilette Chávez de Mira** por brindarme sus consejos cuando los necesité. Por tratar de brindarme su experiencia para sobreponerme a las vicisitudes de la vida.

A mis tíos: **Benjamín y María del Socorro** por su apoyo y sus consejos cuando los necesité. Gracias por toda la ayuda que me brindaron.

A todos mis **otros familiares** por sus oraciones, consejos, críticas, apoyo y buenos deseos, pues directa o indirectamente fueron de gran ayuda.

A **La FAES** por darme mi primera oportunidad laboral, a todas las personas que me brindaron su amistad y apoyo a lo largo de mi proceso de estudio y elaboración de mi trabajo de grado.

A todos mis amigos que me han brindaron su amistad, tiempo y cariño sincero, por compartir su amistad conmigo y permitirme formar parte de sus vidas. Gracias por su apoyo y sus palabras de aliento cuando más las necesité. No escribo sus nombres porque son demasiados pero espero que todos identifiquen con estas breves líneas.

A mis compañeros de tesis por haber aportado su granito de arena para lograr culminar este proceso, que terminó de la mejor manera posible. A su familia gracias por su hospitalidad.

¡GRACIAS!
Luis Antonio Chávez Olivares

Agradezco primeramente a Dios Todopoderoso quien siempre fue mi ayudador y siempre estuvo a mi lado como poderoso gigante para ayudarme, consolarme y darme fuerzas, ánimos e inteligencia en los momentos que más los necesite.

También agradezco a mis padres Luis Ángel Clavel Salazar y Gloria Amanda Toledo de Clavel quienes siempre me dieron su apoyo y ayuda incondicional en todos los aspectos de mi vida hasta el día de hoy, de los cuales tengo un ejemplo a seguir durante todo el transcurso de mi vida.

También agradezco a mis hermanas Roxana Carolina y Gloria Esperanza Clavel las cuales al igual que mis padres me apoyaron y me animaron a seguir adelante en los momentos difíciles, al igual agradezco a mi hermano Ángel Armando Clavel el cual fue de mucha ayuda a nivel académico siendo el mejor docente que tuve así como con palabras de consejo al momento de tomar decisiones.

Al igual es de hacer mención a todos mis compañeros de estudio con los cuales estuvimos juntos apoyándonos unos a otros para alcanzar la meta trazada y llegar hasta el final como hoy en día, compañeros que nunca voy a olvidar a los cuales también les deseo éxitos en sus estudios y en las otras áreas de sus vidas.

Antes de terminar hago mención de todas aquellas personas que de una forma u otra contribuyeron a alcanzar esta meta y estuvieron siempre pendientes brindándome su apoyo y ayuda en los momentos que fue necesario.

Y de esta forma, con la culminación de esta carrera doy fe que con la ayuda de Dios todo se puede y que no hay nada imposible para El, el cual nos ayuda y nos da fuerzas cuando no las hay.

¡GRACIAS!
Luis Antonio Clavel Toledo

A **Dios todopoderoso** por regalarme todo lo que necesité, por ser mí ayuda y apoyo en todo momento, por protegerme, por darme una segunda oportunidad de vida y por regalarme la oportunidad de conocer a tantas personas que de una u otra forma me ayudaron a lo largo de mi carrera.

A mi madrecita **Consuelo Castaneda Vda. de Vásquez** quien no solamente se sacrifico por darme más de lo que necesitaba económicamente. También gracias por su confianza, apoyo, consejos, por adaptarse a mi estilo de vida y soportarme tanto cansancio, mal humor y desatención, así como también todas las noches que ella se desveló junto a mí en el desarrollo de este trabajo de grado.

A mi padre **Guillermo Antonio Vásquez** quien ya no está entre nosotros pero que desde el cielo sé que estuvo velando por mí en todo momento.

A mis **hermanos** por darme su apoyo moral durante toda mi carrera.

A mis **sobrinitos** por distraerme y recordarme que también hay tiempo para jugar y divertirse, no solo para el trabajo.

A todos mis **otros familiares** por sus oraciones, consejos, críticas, apoyo y buenos deseos, pues directa o indirectamente fueron de gran ayuda.

A mi novia **Rosa Delmy Vidal Hernández**, quien ha sido un gran apoyo durante parte final de mi carrera así como también en el desarrollo de este trabajo de grado, por soportarme con mi mal humor y desatención hacia ella. A su familia gracias.

A mis **compañeros de tesis**, por su confianza, ayuda, y consideración. A su familia gracias por su hospitalidad.

A cada una de esas personas tan importantes en mi vida que me brindaron su amistad y cariño sincero, que me enseñaron muchas cosas y me ayudaron de diversas formas en cuanto estaba a su alcance. Gracias a mis excompañer@s, amig@s y docentes de la Universidad, y a otras personas a quienes siempre llevo en mi corazón. Si no escribo sus nombres es porque son demasiados pero se que cuando lean esto se sentirán identificados.

*¡GRACIAS!
Guillermo Rafael Vásquez Castaneda
(MeMuX)*

TABLA DE CONTENIDO

CAPITULO I.GENERALIDADES DEL PROYECTO	11
1.1 INTRODUCCIÓN	XII
1.2 OBJETIVOS	13
1.3 JUSTIFICACIÓN	14
1.4 ALCANCES	16
1.5 LIMITANTES	16
1.6 MARCO HISTORICO.....	17
1.7 ESTRUCTURA ORGANIZATIVA DEL CEMENTERIO NACIONAL SANTA ISABEL.....	18
1.8 SERVICIOS QUE PRESTA EL CEMENTERIO NACIONAL SANTA ISABEL.....	19
1.9 PLANTEAMIENTO DE PROBLEMA.....	19
CAPITULO II. ANÁLISIS DE LA SITUACIÓN ACTUAL	22
2.1 TÉCNICAS DE INVESTIGACIÓN.....	23
2.1.1 ANÁLISIS DE LA SITUACIÓN ACTUAL.....	24
2.1.1.1 RECURSO HUMANO	24
2.1.1.2 EQUIPO INFORMÁTICO.....	27
2.2 ANÁLISIS DE LA INFORMACIÓN RECOLECTADA.....	28
CAPITULO III. ANÁLISIS DE REQUERIMIENTOS DEL PROYECTO	30
3.1 DEFINICIÓN DE REQUERIMIENTOS	31
3.1.1 REQUERIMIENTOS DE INFORMACIÓN.....	31
3.1.2 REQUERIMIENTOS OPERACIONALES.....	34
3.1.3 REQUERIMIENTOS DE DESARROLLO.....	39
3.1.4 REQUERIMIENTOS ECONÓMICOS.....	41
3.2 SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN Y GESTOR DE BASES DE DATOS PARA EL DESARROLLO DEL SISTEMA.....	41
3.3 ESTUDIO DE FACTIBILIDAD	42
3.3.1 FACTIBILIDAD TÉCNICA	42
3.3.2 FACTIBILIDAD ECONÓMICA.....	46
CAPÍTULO IV.- DISEÑO Y DESARROLLO DEL SISTEMA DE INFORMACIÓN	52
4.1 ESTÁNDARES PARA ARCHIVOS, CAMPOS Y VARIABLES.....	53
4.2 SISTEMA DE INFORMACIÓN	56
4.2.1 DIAGRAMA DE CASOS DE USO	56
4.2.2 DIAGRAMAS DE CLASES.....	74
4.3 DISEÑO DE INTERFACES DE ENTRADA Y SALIDA DE DATOS.....	83
4.4 DISEÑO DE LA BASE DE DATOS.....	107
4.4.1 CREACIÓN CONTENEDORES DE INFORMACIÓN (CREACIÓN DE TABLAS).....	107
4.4.2 NORMALIZACIÓN DE LA BASE DE DATOS.....	113
CAPÍTULO V.- PLAN DE IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN.....	115
5.1 OBJETIVOS DEL PLAN DE IMPLEMENTACIÓN	116
5.2 LISTADO DE ACTIVIDADES DEL PLAN DE IMPLEMENTACIÓN	116
CAPITULO VI. DOCUMENTACIÓN DEL SOFTWARE.....	121
6.1 MANUAL DE USUARIO DEL SISTEMA DE INFORMACIÓN	122
6.2 MANUAL DE ADMINISTRACIÓN DEL SISTEMA DE INFORMACIÓN.....	232
6.3 DICCIONARIO DE DATOS	386

CAPITULO VII. CONCLUSIONES Y RECOMENDACIONES.....	397
7.1 CONCLUSIONES	398
7.2 RECOMENDACIONES.....	399
7.3 BIBLIOGRAFÍA	400
7.4 ANEXOS.....	401

CAPITULO
I.GENERALIDADES DEL
PROYECTO

1.1 INTRODUCCIÓN

En el presente documento se detallan todas las partes que conforman el proyecto de trabajo de graduación denominado **“IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DE INFORMACIÓN EN EL CEMENTERIO SANTA ISABEL”**.

El cementerio Santa Isabel es una institución pública de importancia para la ciudad de Santa Ana, ya que la misma se encarga de avalar todas las defunciones del departamento, lo cual es de vital importancia ya que permite mantener la sanidad del municipio y el levantamiento de información para la alcaldía, pero para poder prestar un servicio de calidad a la población necesita poder aplicar los avances tecnológicos con que se cuentan en la actualidad para poder realizar su trabajo de forma eficaz y eficiente.

Por lo que es importante mencionar que actualmente se ha incrementado la demanda de los servicios que el cementerio Santa Isabel provee, como consecuencia de este hecho el volumen de información que se maneja ha aumentado considerablemente y si a esto se le suma toda la información que se tiene desde el año de 1897 de forma manual, a través de libros, entonces podemos decir que se torna difícil el manejo de la misma, situación que es agravada porque no se ha podido desarrollar un sistema de información, debido a que el departamento de informática de la alcaldía de Santa Ana se encuentra empeñado en otras aplicaciones que generan aún más demanda.

Debido a lo anterior, es que este proyecto es de gran importancia, ya que ayudará a mejorar los procesos internos de la institución así como la eficiencia, eficacia y productividad de los empleados dando como resultado un mejor servicio para la población y el municipio.

1.2 OBJETIVOS

OBJETIVO GENERAL

- Implementar un sistema de información para la mecanización de los procesos realizados para brindar los diferentes servicios que presta el cementerio municipal Santa Isabel de Santa Ana.

OBJETIVOS ESPECIFICOS

- Recopilar los mecanismos de funcionamiento de los procesos utilizados en el cementerio Santa Isabel.
- Clasificar e interpretar los datos recopilados acerca de los procesos, para ser considerada su automatización y disminuir el tiempo de espera en la atención de los usuarios del servicio.
- Reemplazar los métodos manuales por métodos automatizados para lograr la agilización de los servicios a través del sistema de información.
- Proveer al cementerio nacional de Santa Ana un método de automatización para la administración de expedientes y procesos administrativos del personal.
- Proteger la integridad de la información y su persistencia a través del tiempo mediante el almacenamiento digital.
- Realizar la implementación y pruebas de la medida de solución en el cementerio Santa Isabel, con los recursos gestionados con la alcaldía municipal de Santa Ana.

1.3 JUSTIFICACIÓN

En el cementerio municipal Santa Isabel Santa Ana, se identificó la necesidad de mejorar los procesos del manejo de la información a nivel interno, debido a que hasta el momento todas las operaciones se hacen de forma manual, haciendo uso de métodos convencionales, lo cual provoca lentitud en el manejo de las mismas. Generando así demoras en la atención al público, así como el retardo del cumplimiento de las tareas contenidas en el plan de trabajo de los empleados al momento de realizar sus labores.

Dicha situación podrá solventarse con el desarrollo de este proyecto, ya que actualmente cuando se solicita información sobre los servicios del cementerio, los empleados deben determinar el libro en el que se encuentra contenida y luego buscar el registro solicitado entre una gran cantidad de información. En muchos de esos casos la información se encuentra deteriorada o inutilizable debido al deterioro de los libros que almacenan la información. Si la información es encontrada y utilizable el cementerio procede a realizar los mecanismos manuales para solventar las solicitudes.

Otro punto muy importante que se destaca es que el cementerio necesita que se pueda garantizar la seguridad de la información, ya que es imprescindible la utilización de los datos desde la apertura del cementerio, por lo que se deben utilizar el almacenamiento digital, debido a que es mucho más seguro para garantizar la persistencia a través de respaldos o copias de seguridad. Situación que actualmente no es posible realizar y esto ocasiona pérdida de información importante para los procesos que se realizan en el cementerio.

Debido a los inconvenientes que se mencionaron anteriormente, se propone el desarrollo de un sistema para el manejo de la información, buscando de esta forma agilizar y automatizar los trámites de los usuarios, así como las operaciones del personal que laboran en dicha institución, permitiendo lograr de esta forma una mayor eficiencia en sus labores.

Las necesidades mencionadas anteriormente han sido expresadas por parte de la administración del Cementerio Municipal Santa Isabel de Santa Ana, quienes actualmente se encuentran gestionando el equipo informático necesario para la modernización de la entidad. En los anexos se presenta una carta en la cual la entidad manifiesta la necesidad existente, y por lo cual se vuelve necesario desarrollar este proyecto como un trabajo de grado para la institución.

Tomando en cuenta que el objetivo más importante de esta institución es la atención de calidad para los ciudadanos que solicitan el servicio, por lo que el tener una herramienta que facilite dichas acciones se ha vuelto una necesidad para el cementerio.

Con el Sistema de Información se pretende lograr que el cementerio cumpla con las siguientes expectativas:

- Mejorar la atención a las personas que hacen uso de los diferentes servicios que el cementerio brinda, esto mediante una disminución en el tiempo de atención al momento de realizar los procesos de papeleo, captura y búsqueda de información.
- Mejorar el manejo de la información al garantizar la integridad de la misma al momento de ser ingresada, procesada y almacenada por parte de los usuarios. Dicha integridad se verá mejorada debido a que esta se almacenara de forma digital y podrá ser respaldada en copias de seguridad al momento que se desea almacenarla en diversos dispositivos de almacenamiento, situación que no es posible con los actuales métodos con que se cuentan, como lo son libros y folios en estado de deterioro.

Cabe mencionar que actualmente no existe ningún método de respaldo de información en caso de haber alguna perdida, ya que únicamente se cuenta con una copia de los datos escritos a mano en libros.

- Llevar un mejor control sobre los cobros, impuestos y servicios a fin de garantizar una mejor transparencia de la institución ante la alcaldía municipal de Santa Ana.

1.4 ALCANCES

Con la presente propuesta de trabajo de grado sobre la **IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DE INFORMACIÓN EN EL CEMENTERIO SANTA ISABEL** se delimitan los siguientes alcances:

1. Establecer los requisitos de hardware y software necesarios para la correcta utilización y funcionamiento del sistema de información.
2. Implementar un Sistema de Información para agilizar y automatizar los procesos del cementerio municipal Santa Isabel.
3. Administración de expedientes del personal del cementerio, por medio del sistema de información y el cumplimiento de su plan de trabajo.
4. Almacenar y administrar la información y cobros para brindar los diferentes servicios disponibles en el cementerio a los familiares de las personas fallecidas.
5. Generar informes en los procedimientos económicos y de la demanda de los servicios utilizados en periodos determinados de tiempo.
6. Elaborar el manual del usuario y el manual de instalación del sistema de información desarrollado.
7. Establecer un plan de implementación del sistema de información desarrollado para el cementerio municipal Santa Isabel.
8. Realizar capacitaciones para el personal con respecto al uso del sistema.

1.5 LIMITANTES

- Falta de cooperación del Departamento de Informática de la Alcaldía Municipal de Santa Ana en la entrega de la información técnica solicitada.
- Falta de presupuesto asignado al Cementerio Santa Isabel para invertir en modernización del equipo.
- Los procedimientos utilizados en el cementerio municipal Santa Isabel, se encuentran sujetos a cambios por parte de los consejos municipales de la alcaldía, así mismo de un posible cambio en la ley y reglamento de los cementerios municipales, ya que esta ley fue aprobada en 1973 y 1977 respectivamente, sufriendo reformas la primera en 1994.

1.6 MARCO HISTORICO

La información que a continuación se detalla fue proporcionada por miembros del personal administrativo del Cementerio Santa Isabel, detallando de forma breve la historia del cementerio desde sus orígenes hasta el tiempo presente.

El municipio de Santa Ana cuenta actualmente con 13 cementerios municipales, siendo el cementerio Santa Isabel el principal y el único que posee administración. A estos cementerios se les denomina periféricos, por encontrarse en la periferia de la ciudad, correspondiendo cada uno a diferentes cantones del municipio de Santa Ana. Así mismo estos cementerios no hacen ningún tipo de cobro o registro de las defunciones y solamente se limitan a brindar el servicio de sepelios, ya que estos procesos son realizados por el cementerio principal.

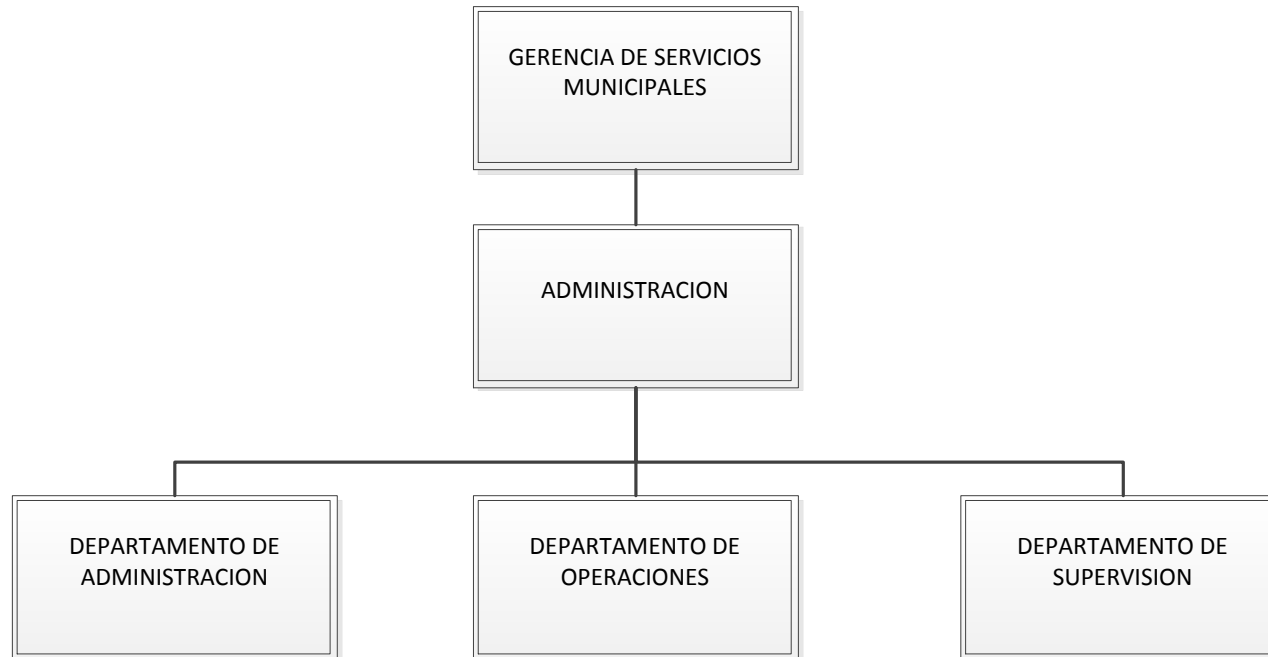
Inicialmente el cementerio nacional de Santa Ana se encontraba donde actualmente se encuentra el Hospital Nacional San Juan de Dios, luego el cementerio fue trasladado donde se encuentra actualmente el Parque Anita Alvarado. Los restos de las “personas importantes” o de la alta sociedad fueron exhumados y trasladados al actual cementerio municipal.

El Cementerio Santa Isabel fue fundado el 1 de enero de 1897 y su primer entierro fue gratis. Territorialmente era pequeño, sin embargo debido a la creciente demanda de los fallecidos a partir de los años 80, fue necesario agrandar el cementerio, teniendo actualmente un tamaño cercano a las 36 manzanas.

El cementerio desde su fundación ha está dividido en 3 categorías: Nichos (mausoleos), fábrica media y fábrica ínfima. Esto se refiere al tipo de lugar para enterrar las osamentas. Así mismo el cementerio se encuentra distribuido en números de puestos, fosas, filas, cuadros y calles. Tal como si fuera la nomenclatura de una dirección.

En la década de 1980 se atendían alrededor de 4040 muertos al año, necesitándose crear dos tomos por año por la alta demanda. En el año 2001 hubo un aproximado de 1200 muertos en el año, ya que el cementerio trabaja los 365 días del año.

1.7 ESTRUCTURA ORGANIZATIVA DEL CEMENTERIO NACIONAL SANTA ISABEL



1.8 SERVICIOS QUE PRESTA EL CEMENTERIO NACIONAL SANTA ISABEL

El cementerio nacional Santa Isabel brinda actualmente los siguientes servicios:

- Inhumación en:
 - Nicho.
 - Fabrica media.
 - Fabrica ínfima.
- Exhumación.
- Pago de impuestos de contratistas de construcción.
- Generación de título a perpetuidad.

1.9 PLANTEAMIENTO DE PROBLEMA

El cementerio actualmente no cuenta con ningún tipo de sistema informático que ayude a llevar a cabo los procesos administrativos que se llevan a cabo en el cementerio, esto conlleva a que el trabajo sea más lento y da cabida a errores en la veracidad de la información, ya que según se entrevistó al personal del cementerio, se han dado errores en el llenado de formularios y al transferir esa información a los libros de registro del cementerio.

En la tabla 1 se muestran los datos pertenecientes a los últimos 5 años sobre la cantidad de personas enterradas en el cementerio:



Tabla 1: Fallecidos anualmente

Se puede observar el incremento a partir del año 2008, lo cual se convierte en una mayor demanda para el cementerio para mantener actualizados sus registros y poder brindar el servicio a la población que lo solicita. Actualmente se entierran un aproximado de 2 a 4 personas a diario, lo que en promedio sería 3 personas enterradas diariamente, por lo que aproximadamente se tendría una demanda de 1008 personas fallecidas, lo que es una demanda considerable, sin embargo debido a violencia que enfrenta nuestro país podemos observar a través del grafico que hasta la fecha el cementerio cuenta con 985 casos atendidos, que según el promedio deberían ser 756 casos, por lo que se presenta una diferencia de 229 casos por arriba de los cálculos de este promedio, lo que representa un aumento del 30% de los casos, por lo que podemos concluir que no es viable manejar la información de las personas fallecidas con los métodos actuales, debido a que se muestra una tendencia al aumento de fallecidos anuales.

En el año 2004 se intentó digitalizar todos los procesos realizados a través de un sistema informático mecanizado, pero no se pudo llevar a cabo debido a que la administración de ese periodo consideró que había proyectos más urgentes en esos momentos.

En lo referente a los pagos de impuestos y fosas, estas se comenzaron a pagar a partir del año 1993, aunque las refrendas anuales por el espacio de entierro se han pagado desde la fundación del cementerio, según se muestra en la tabla 2:

Período	Fosas	Impuestos
1993-1994	\$ 01.37	\$ 01.37
1995-1996	\$ 02.75	\$ 06.86
1997-2010	\$ 06.00	\$ 12.00

Tabla 2: Diferentes pagos

Cabe mencionar que todos los impuestos son anuales a excepción de las refrendas que se realizan cada 7 años, es decir que es un tributo que se paga para garantizar que se mantiene el derecho de utilizar el espacio de terreno por el que se ha pagado, ya sea fabrica media o mínima, nichos o fosas comunes. De 1987 al 2007 se pagaba un tributo de \$8.83, pero a partir del 2007 a la actualidad se cancelan por dichas refrendas \$25.75.

También existen otros aranceles diferentes para el cementerio privado existente en la ciudad y los cementerios periféricos, según lo muestra la tabla 3:

Lugar	Impuesto
Parque Jardín Las Flores	\$ 41.20
Cementerios municipales periféricos	\$ 06.18

Tabla 3: Impuestos

El cementerio Santa Isabel no realiza administración, ni servicios para los cementerios periféricos o el cementerio privado, solamente se limita al cobro de los impuestos por derecho de entierro de las personas que allí son enterradas, sin embargo no registra en sus libros la existencia de estas personas, ya que la institución se limita a llevar un control de los espacios locales.

CAPITULO II. ANÁLISIS DE LA SITUACIÓN ACTUAL

2.1 TÉCNICAS DE INVESTIGACIÓN

Para el desarrollo de este proyecto se han empleado diferentes técnicas de investigación entre las cuales se pueden mencionar:

OBSERVACIÓN DIRECTA.

Esta técnica se basó en la realización de visitas al lugar de campo, lo cual se hizo en diferentes ocasiones permitiendo de esta forma observar y analizar las operaciones que ahí se desarrollan, así como las herramientas de trabajo empleadas, cantidad de personal y la atención al cliente que se presta.

REVISIÓN DOCUMENTAL.

Consistió en la consulta de libros, informes, documentos, archivos y otros materiales que aportaron información para comprender y delimitar el problema a tratar.

ENTREVISTA.

Con esta técnica de investigación se logró recopilar la información de una forma más directa, pues fue posible obtener de primera mano los puntos de vista de las personas directamente involucradas en los procesos, es decir en el lugar del problema o necesidad.

Para el caso de dicha actividad se procedió a entrevistar al personal administrativo del cementerio incluido el administrador y su asistente, así como a un trabajador encargado de las sepulturas, además se entrevistó al jefe del departamento de informática de la alcaldía municipal.

Con este método se pudieron obtener las principales necesidades que actualmente se tienen, ya que dichas necesidades fueron expresadas directamente por el personal que labora en el lugar, escuchando así sus principales puntos de vista, así como la necesidad que ellos aseguran tener sobre una mejora en los métodos actuales de trabajo, corroborando con ello la necesidad que actualmente se tiene sobre el manejo de la información.

En el caso del personal entrevistado de la alcaldía municipal, se pudo saber el equipo de cómputo con el que actualmente se dispone, permitiendo así tener un panorama más amplio al momento de tomar una decisión sobre el software y hardware a utilizar.

2.1.1 ANÁLISIS DE LA SITUACIÓN ACTUAL

Para el desarrollo del sistema, es necesario hacer un estudio previo de la situación en la que actualmente se encuentra el cementerio Santa Isabel, para lo cual detallamos a continuación las condiciones en las que se encuentra dicha institución en la actualidad.

2.1.1.1 RECURSO HUMANO

Actualmente el RRHH con que se dispone en el cementerio Santa Isabel asciende a un total de 25 trabajadores, conformando así un total de 11 secciones.

A continuación se mencionas de forma breve las descripciones de cada uno de los puestos así como las funciones que estos realizan:

EXCAVADORES (EQUIPO DE 5 PERSONAS)

Se encargan de abrir diariamente las sepulturas que no han sido refrendadas después de 7 años y que serán utilizadas por la demanda diaria de inhumaciones (entierros) nuevos. Trabajan desde las 5 de la mañana y cada uno de ellos tiene la responsabilidad de abrir una sepultura por día. Los excavadores también se encargan de atender exhumaciones que han sido solicitadas por el público y que corresponden a restos que ya tienen más de 7 años. Cuando hay suficientes sepulturas abiertas, ellos realizan otras tareas como la chapoda y la limpieza de ripio.

ENTERRADORES (EQUIPO DE 6 PERSONAS)

Estas personas tienen la responsabilidad de asistir las inhumaciones, es decir, los entierros diarios y acordes a la demanda, realizar exhumaciones, entierro de óbitos (restos hospitalarios), enseñar ubicaciones al público que lo solicite y relacionadas a sepulturas.

ALBAÑILES Y AUXILIARES (EQUIPO DE 4 PERSONAS)

Son responsables de asistir los entierros en nichos y mausoleos, abrir y cerrar colas de los nichos, realizar exhumaciones en nichos, cooperar en el mantenimiento de las instalaciones.

OFICIOS VARIOS (EQUIPO DE 3 PERSONAS)

Son los encargados de realizar la limpieza de calles garantizando así la retirada de hojas y basura de forma diaria y permanente, enseñar ubicaciones, apoyar a cualquiera de los equipos de trabajo, apoyar cualquier otra tarea que la administración les asigne.

CUSTODIO (UNA PERSONA)

Es la persona que garantiza que el cementerio funcione en los fines de semana, asueto y vacaciones, responsable de hacer el listado, previa investigación de campo y de libros en la oficina, de las sepulturas que serán exhumadas por mora después de 7 años o más de haber sido inhumados. Atención al público fines de semana, vacaciones y asueto.

SUPERVISOR (UNA PERSONA)

Se encarga de supervisar el trabajo que realizan los contratistas, garantizando que sean respetados las líneas y medidas, los materiales presupuestados y la calidad de la obra, asegurando que no se dañen en el proceso de construcción otras estructuras ni que se construya en el lugar equivocado. Ayuda a supervisar a los azadoneros y regadoras para que respeten el campo santo, paguen sus impuestos acorde a los trabajos que realizan e informe a la administración de cualquier anomalía relacionada con contratistas, azadoneros y regadoras.

JARDINERO (UNA PERSONA)

Cuida las zonas verdes, jardines y vivero. Abono, combate de plagas, chapodas, riego y la presentación estética de los jardines.

ORDENANZA (UNA PERSONA)

Apoyar a la administración con el aseo de oficinas y corredores, llevar y traer documentos en las diferentes oficinas y departamentos de la Alcaldía Municipal, entregar correspondencia a cualquier organización, institución de parte de la administración, apoyo al equipo administrativo.

AUXILAR DEL CEMENTERIO - SECRETARIA ADMINISTRATIVA (UNA PERSONA)

Responsable de los títulos de propiedad, documentos varios, hacer los listados de ingresos diarios, mensuales y anuales, atención al público, apoyo al administrador y al

equipo administrativo, control de incapacidades y tiempo compensatorio, atender las llamadas.

AUXILAR DEL CEMENTERIO - ATENCION AL PÚBLICO

Atender al público directamente en la información que requieran, el cobro de impuestos por inhumación, exhumación, traslado de restos, cobrar impuestos por derecho a construcción, puesta de lápidas y cruces, por licencia anual a contratistas, impuesto de agua a regadoras, cobro de refrendas y venta de propiedad, colectar el dinero que ingresa al cementerio, realizar cortes y entregas de los mismos llevando un control diario del dinero recaudado. Debe de cuidar de la entrega de recibos de Fórmula 1 ISAM para cada una de los cobros que realiza, llenar las hojas de asentamientos de entierro de forma correcta sin alterar ni omitir ningún dato e información.

ADMINISTRADOR (UNA PERSONA)

Responsable de coordinar toda la gestión que se realiza en el cementerio, elaborar, implementar y monitorear el plan anual de trabajo con su debido presupuesto, atender amablemente cualquier problema del público, coordinar el trabajo de todo el personal de campo y administrativo. Realizar las gestiones necesarias para el cumplimiento de metas y objetivos. Garantizar el respeto a los derechos y dignidad de los trabajadores estableciendo relaciones laborales sanas y justas. Mantener una adecuada coordinación y comunicación con las autoridades municipales, en especial con la Gerencia de Servicios Municipales.

Como se pudo observar, el cementerio está conformado por 11 secciones realizando tareas completamente distintas y bajo diferentes entornos de trabajo.

Para este caso, el sistema de información será utilizado por la secretaria administrativa, el auxiliar de administración y por el administrador mismo, es decir por todas aquellas personas que prestan atención al cliente, realizan trámites administrativos y elaboran documentos.

2.1.1.2 EQUIPO INFORMÁTICO

Actualmente en cementerio Santa Isabel Santa Ana, se cuenta con un total de dos estaciones de trabajo (computadoras), conformando así todo el equipo de cómputo con que se dispone en la actualidad.

Dichas computadoras se encuentran ubicadas en la oficina administrativa, cuyo fin es elaborar documentos o cartas que son enviadas a la alcaldía Municipal u otros destinos. Cabe mencionar que dichos equipos no se utilizan para ningún procedimiento de trabajo de la institución para servicios de atención al público,

De ambos equipos únicamente se encuentra operando uno de ellos, ya que el segundo se encuentra averiado y aun no se ha determinado la causa del problema, por ende únicamente se dispone de una estación de trabajo para el uso mencionado anteriormente.

Otro aspecto muy importante son las capacidades que estos poseen, tanto a nivel de hardware como a nivel de software. Dichas características se describen en la tabla 4:

Equipo I (En buen estado)

Hardware	Software
Microprocesador Pentium IV 1.6Hz	Microsoft Windows XP
Memoria RAM 1Gb	Microsoft Office 2003
Disco duro 80Gb	
Unidad CD/DVD-RW	

Tabla 4: Características de equipo Existente I

Como se pudo observar en la tabla 4, dicho equipo posee características aceptables para el uso de oficina, tanto a nivel de hardware como a nivel de Software respondiendo de forma aceptable al uso de parte de los usuarios.

Con respecto al segundo equipo (computadora averiada) se puede decir que este posee características de hardware obsoletas, no cumpliendo así los requerimientos mínimos necesarios para la instalación y uso de la mayoría de programas actuales.

En cuanto al Software que este posee también se puede decir que se encuentra desactualizado, contando con versiones antiguas a nivel de sistema operativo y Office.

Dichas características se describen en tabla número5:

Equipo II (PC averiada)

Hardware	Software
Microprocesador Pentium III 850Hhz	Microsoft Windows 2000
Memoria RAM 128Mb	Microsoft Office 2000
Disco duro 10Gb	
Lector CD-ROM	

Tabla 5: Características de equipo Existente II

Como se pudo observar en la tabla 5, dicho equipo prácticamente se considera inutilizable primeramente por el problema a nivel físico que presenta y luego por las características a nivel de Hardware y Software que este posee.

En la tabla número 6 se muestran las características ideales a nivel de hardware y software que un equipo promedio debe poseer para realizar trabajos de oficina así como para ejecutar aplicaciones que trabajan bajo entornos de escritorio.

Hardware	Software
Microprocesador Intel Core2Duo 2.4Ghz	Microsoft Windows 7 profesional
Memoria RAM1Gb	Service Pack 1
Disco duro 160Gb	Microsoft Office 2010
Unidad CD/DVD RW	Internet Explorer 8
Interfaz de red Fast Ethernet 100Mbps	Software Antivirus
Puertos USB	

Tabla 6: Características de equipo

El equipo descrito en la tabla 6 se considera recomendable para un entorno de oficina ya que con dichas características de Hardware y Software se garantiza al usuario un desempeño mucho más eficiente, permitiendo la ejecución de múltiples tareas de forma simultánea sin percibir lentitud o demoras en los tiempos de respuesta por parte del usuario.

2.2 ANÁLISIS DE LA INFORMACIÓN RECOLECTADA

Mediante el uso de las técnicas de investigación mencionadas anteriormente como lo son observación directa, entrevistas y consultas bibliográficas, se pudieron identificar los procedimientos de trabajo, los servicios que el cementerio presta, el equipo informático con que se dispone y el RRHH que este posee.

Información obtenida por observación directa

Mediante la observación de campo se pudieron verificar muchos aspectos del lugar como por ejemplo la infraestructura física, mobiliario y equipo, instalación eléctrica, equipo informático, herramientas de trabajo, etc.

En lo referente a la infraestructura física se pudo determinar el estado en que esta se encuentra, la distribución de sus departamentos así como el mantenimiento que esta recibe, concluyendo que actualmente esta se encuentra en malas condiciones debido a que es una edificación antigua, construida con materiales que actualmente no cumplen con las normas mínimas de calidad como por ejemplo adobe y madera.

Con respecto a la instalación eléctrica se hicieron algunas observaciones en cuanto a su estado, siendo este poco eficiente ya que carece de medidas de protección como lo son polarización, pararrayos, planta generadora en caso de un corte de energía, cableado en mal estado, mala distribución de tomas, así como construcción sin ninguna norma de diseño. Considerándose esta insegura para la conexión de equipos eléctricos.

En lo que toca que ver con el equipo informático se concluyó que únicamente se cuenta con 2 computadoras, una de las cuales se encuentra averiada y en desuso, y el segundo equipo se utiliza para la elaboración de documentos de texto y hoja de cálculo.

Ahora bien, mediante la revisión documental y entrevistas se pudo observar la forma en que los datos son manejados y almacenados por parte de los empleados. Como se mencionó anteriormente todos los procesos que tocan ver con recopilación y manejo de la información son completamente manuales, siendo el principal medio de almacenamiento folios y libros que en la mayoría de los casos se encuentran en parcial o total estado de deterioro.

En lo que respecta a las entrevistas, se entrevistó al personal administrativo incluyendo al administrador del cementerio y a su asistente, así como también al jefe del departamento de informática de la alcaldía municipal.

Con dichas entrevistas se pudo obtener información más detallada sobre los procesos internos, más que con las observaciones iniciales, ya que el administrador explicó cada uno de los procedimientos de trabajo así como el tipo de documentación que se maneja y el método de ejecución del mismo.

CAPITULO III. ANÁLISIS DE REQUERIMIENTOS DEL PROYECTO

3.1 DEFINICIÓN DE REQUERIMIENTOS

Este apartado muestra los diferentes requisitos necesarios para diseñar, desarrollar e implementar el sistema de información que resuelva las necesidades de la obtención de información veraz al instante en el Cementerio Municipal Santa Isabel. Para determinar estos requerimientos de manera ordenada, se ha dividido este análisis de la siguiente manera:

- 1.- Requerimientos Funcionales
- 2.- Requerimientos Operativos
- 3.- Requerimientos de desarrollo

3.1.1 REQUERIMIENTOS DE INFORMACIÓN

Estos definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

A continuación se presentan los requerimientos funcionales para cada uno de los sistemas contemplados en el trabajo de grado:

1. Sistema de Información

Generalidades del Sistema

- El sistema será capaz de trabajar en la red interna que se diseñara en el Cementerio Municipal Santa Isabel.
- El sistema debe proporcionar ayuda al usuario en caso de ser requerida.
- El sistema deberá digitalizar los procesos seleccionados en el análisis de la situación actual.

Proceso de autenticación de Usuarios

- El sistema debe poseer roles de usuario.
- El proceso de autenticación a través de los roles de usuario que sean asignados debe permitir ver al usuario solamente la información pertinente.
- El sistema debe poseer una cuenta de administración de usuarios.

- El sistema debe llevar un registro de las transacciones realizadas por los diferentes usuarios.
- La autenticación de usuarios se debería hacer en dos niveles, el primer nivel de validación será a través de la autenticación de usuarios en el directorio activo con los permisos generados por el administrador del sistema, posteriormente se autenticará el usuario en el sistema para poder acceder al mismo, lo que determinara que actividades puede hacer el usuario con el sistema. Sin embargo debido a la falta de recursos solamente se trabajar con la capa de autenticación de la aplicación.

Administración de Usuarios

- El sistema debe proveer los medios necesarios para la creación, actualización. No se contempla la eliminación de usuarios como tal, ya que debido al registro de transacciones y el manejo de elaboración de cobros el sistema no puede borrar usuarios sino hacer cambios de estados para dar altas o bajas de usuarios.
- El sistema permitirá la creación de usuarios, y los niveles de permisos serán asignados a través del sistema. Sin embargo debería ser a través del directorio activo y el sistema a desarrollar.

Cobro de aranceles

- El sistema debe permitir dar mantenimiento (creación y modificación) del catálogo de aranceles que se cobran para los diferentes servicios que se brindan en el cementerio.
- El sistema debe generar el registro de cobros de los diferentes servicios prestados por la institución e imprimir la respectiva formula ISAM.

Elaboración de actas

- A partir de la validación de los pagos de los diferentes aranceles cobrados para la prestación de un determinado servicio, el sistema deberá permitir elaborar un acta con todos los datos del contratante del servicio para dar constancia del servicio contratado.

Elaboración de títulos de propiedad

- A partir de la validación de los pagos de los diferentes aranceles cobrados para la prestación de un determinado servicio, el sistema deberá permitir elaborar un título de compra.

Calendarización de actividades del plan de trabajo

- El sistema a través de los datos introducidos por el administrador del cementerio o a quien delegue este debe ser capaz de generar en un rango de fechas definido la calendarización de las diferentes actividades a realizarse.
- El sistema debe permitir añadir o modificar eventos a realizarse.
- El sistema debe generar un calendario de eventos para ser impreso.
- El sistema debe generar un censo diario con las actividades propuestas para cada día, según el plan de trabajo elaborado. Así mismo al final del día deberá permitir registrar el cumplimiento de las actividades ejecutadas.

Expedientes del personal

- El sistema deberá permitir realizar expedientes de los usuarios registrados en el sistema, registrando datos personales, permisos laborales, turnos, labores asignadas y amonestaciones.
- El sistema deberá permitir la administración de los expedientes laborales (creación y modificación). Así como la baja de personal que deje de laborar en la institución.

Informes

- El sistema debe generar el informe de ingresos diarios para ser remitidos al departamento de contabilidad de la alcaldía municipal. (impreso y digital)
- El sistema deberá generar un informe con las transacciones registradas a lo largo de un determinado periodo para la elaboración de libros que exige la ley general de cementerios que se mantengan en la institución.

3.1.2 REQUERIMIENTOS OPERACIONALES

Son aquellos requerimientos necesarios para el óptimo funcionamiento del sistema, según detalle:

Legales

El software tiene requerimientos operativos legales que cumplir, los cuales son los Derechos de Autor, este derecho se encuentra regulado por la Ley de Fomento y Protección de Propiedad Intelectual, en la cual Art. 32 asevera que en los programas de ordenador están protegidos los Derechos de Autor, y se encuentran incluidos en el régimen de protección del capítulo II Art. 13 de la referida ley.

Otra ley que regula los derechos de autor se encuentra en el Reglamento General de Procesos de Graduación de la Universidad de El Salvador, donde el Art. 29 establece, que son propiedad exclusiva de la Universidad los trabajos de graduación y solo ella puede disponer de los mismos, y autorizar a otros para que puedan hacer uso de dichos trabajos.

En el país existe una entidad reguladora de las copias ilegales de software la cual es Bussines Software Alliance (BSA), esta institución presenta una denuncia a la Unidad de la Propiedad Intelectual de la Fiscalía General de la Republica, para que ella se encargue de realizar el proceso de aprehensión de las personas u organizaciones que cometen el delito contra la propiedad intelectual. Este delito es penalizado con 4 años de prisión, de acuerdo con el Art. 227 del Código Penal. Sin embargo la legislación establece una salida alternativa de la conciliación (pago a cambio de cárcel).

La alcaldía Municipal de Santa Ana deberá gestionar la compra del software para poder adquirir los derechos de utilización sobre el mismo, estos derechos serán pactados en la solicitud pudiendo incluirse derechos sobre código fuente.

Software

Para el desarrollo del proyecto se requiere el uso del siguiente software de manejo de oficina, desarrollo y administración de base de datos.

Software requerido para el desarrollo del sistema

Software	Etapas de Desarrollo del sistema		
	Situación Actual	Análisis y diseño	Programación, prueba y documentación
Microsoft Word	x	x	x
Microsoft Excel	x	x	x
Microsoft Visio	x	x	x
Microsoft Visual Basic 2010		x	x
Microsoft SQL Server 2008 R2 Express		x	x

Tabla 7: Software requerido

Características necesarias para su funcionamiento.

- 1.- Framework. Este será la base sobre el cual estará montado el sistema, este reúne un conjunto de lenguajes y servicios sobre los cuales es ejecutado el proyecto.
- 2.- Sistema operativo de estaciones de trabajo. Se recomienda un Windows XP SP3 o superior, según las ofertas actuales en el mercado.
- 3.- Un espacio mínimo en disco duro para instalación de 160GB.
- 4.- Gestor de base de datos. Se recomienda el gestor de base de datos Microsoft SQL Server 2008 R2 Express, debido a la integración que existe con la plataforma .NET.

Hardware

A continuación se describen las características del equipo informático requerido para la implementación del sistema de información.

2 ESTACIONES DE TRABAJO DELL OPTIPLEX 380

No	COMPONENTES	DESCRIPCIÓN
1.-	PROCESADOR	Intel® Pentium® E5400 (2MB Caché, 2.70GHz, 800MHz FSB)
2.-	SISTEMA OPERATIVO	Windows® 7 Profesional de 32 bit con Medio en Español
3.-	MEMORIA	Memoria de 2GB Dual Channel DDR3 SDRAM 1333MHz - 2DIMMs
4.-	MONITOR	Monitor Dell E1910H - Plano y Pantalla Ancha de 18.5 pulgadas
5.-	TARJETA DE VIDEO	Integrated Video, Intel® GMA X4500
6.-	DISCO DURO	Disco Duro de 250GB Serial ATA (7200RPM) con DataBurst Caché™
7.-	DISPOSITIVO ÓPTICO	Single Drive: 16X (DVD+/-RW) Burner Drive
8.-	TECLADO DELL	Dell Multimedia Teclado - Español
9.-	MOUSE DELL	Dell Laser Mouse - Black Gloss
10.-	LECTOR DE MEMORIA	Lector de Tarjeta de Medios Dell 8-in-1
11.-	BOCINAS DELL	Bocinas Dell AX210 Estéreo USB
12.-	MODEM	56K PCI Data Fax Modem
13.-	TARJETA DE RED	Intergrated PCIE 10/100/1000
14.-	GARANTÍA	Un Año de garantía limitada. Servicio en el sitio con respuesta el siguiente día laborable.

Tabla 8: Características de equipo

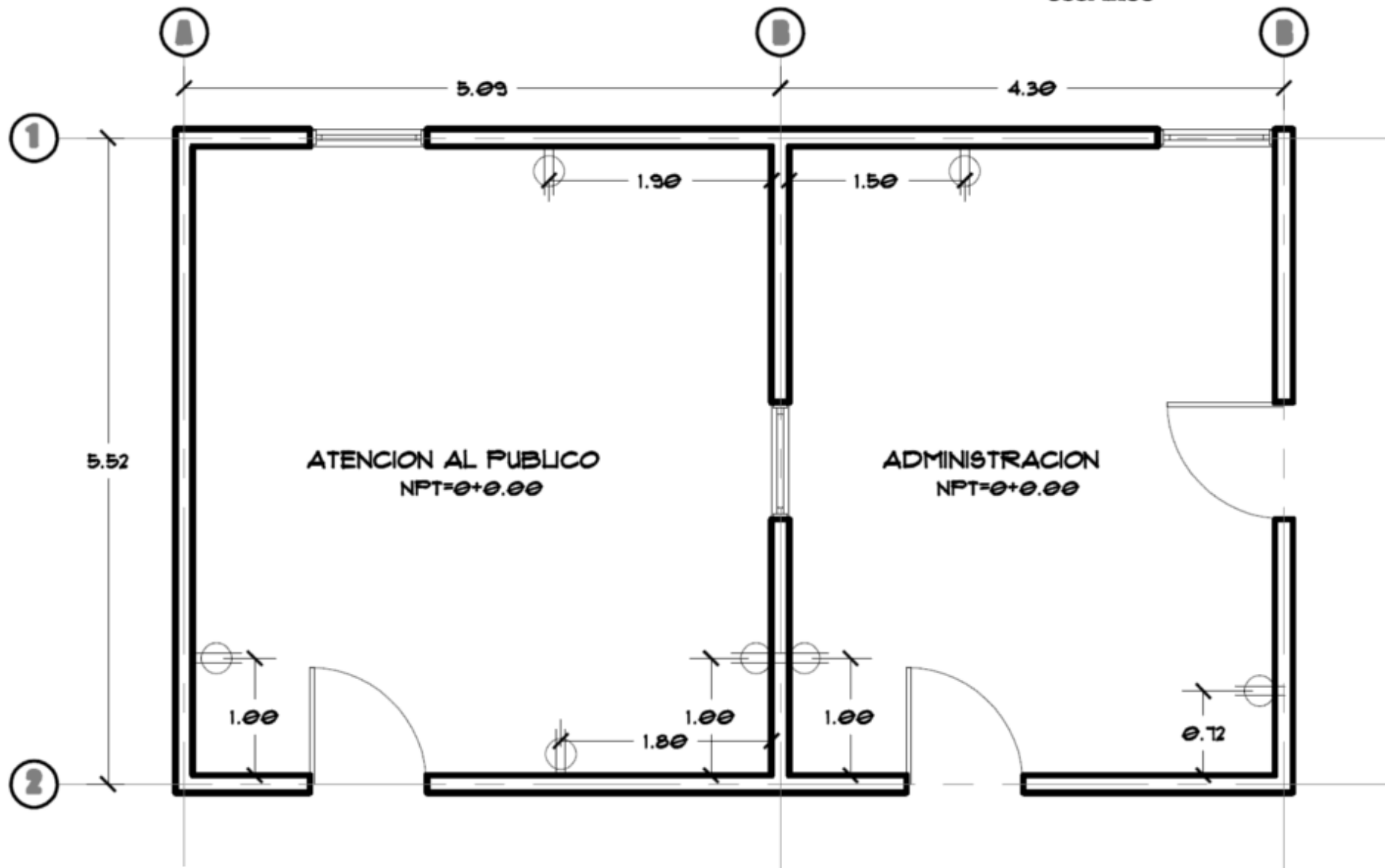
Distribución en planta de oficinas del Cementerio Municipal Santa Isabel

Uno de los aspectos mas importantes para la operación del sistema es la forma en la que se han distribuido los diferentes equipos pensando en la comodidad del usuario así como de las necesidades que se presenta en el lugar, cada componente que se presenta en el diagrama siguiente ejemplifica la forma que en que estarán dispuestos las estaciones de trabajo, así como las ubicaciones sugeridas de los puntos de red necesarios para la utilización del sistema en la red local.

El diagrama siguiente presenta la distribución en planta de las oficinas del Cementerio Municipal Santa Isabel:

PLANTA DE TOMACORRIENTES EXISTENTE

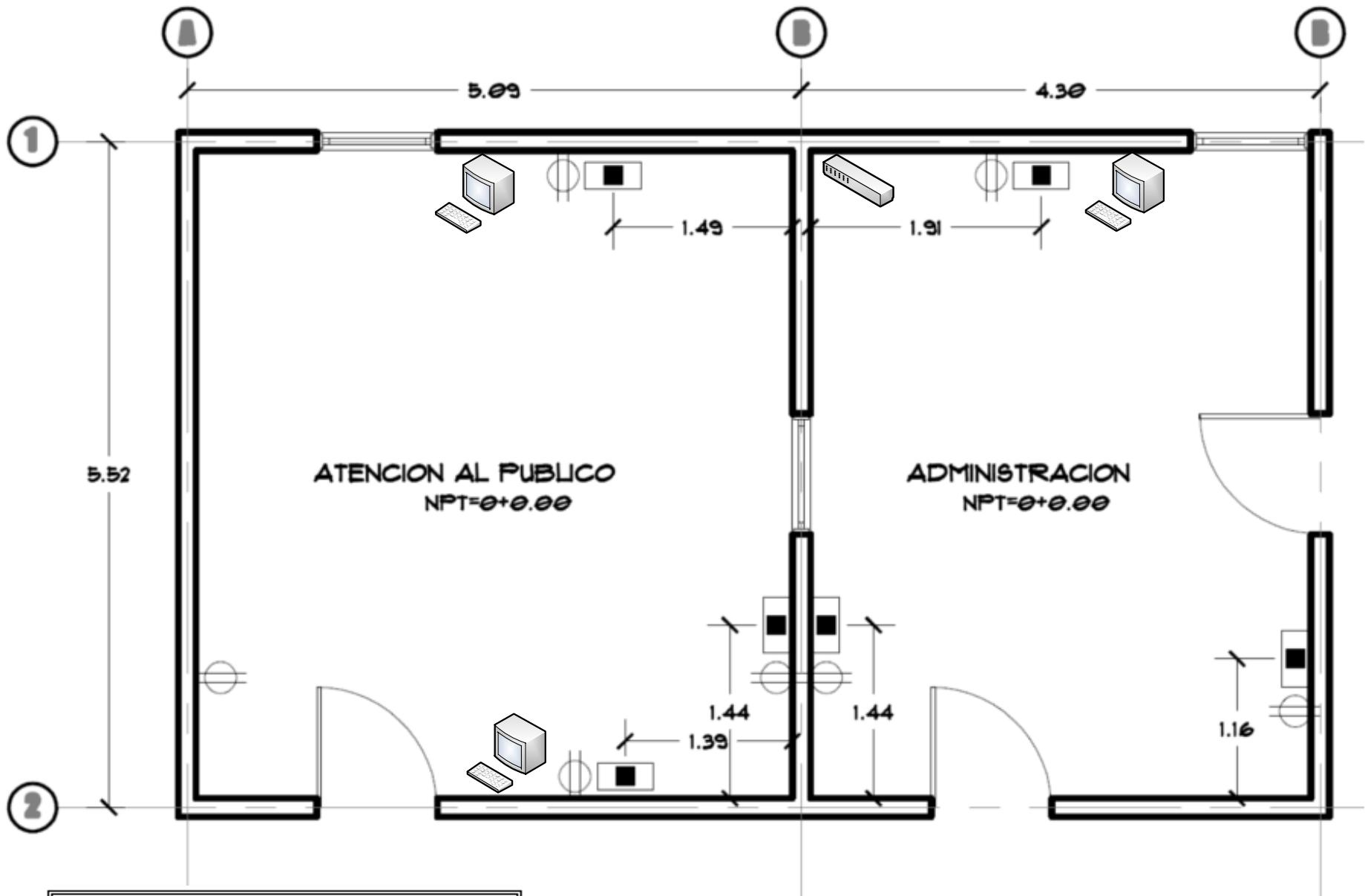
esc. 1:100



SIMBOLOGIA	
ID	DESCRIPCION
	TOMA CORRIENTES EXISTENTES

PLANTA DE TOMACORRIENTES Y RED DE DATOS PROPUESTA

esc. 1:100



SIMBOLOGIA	
ID	DESCRIPCION
⊕	TOMACORRIENTES DOBLES POLARIZADOS PROPUESTOS
■	PUNTOS DE RED DE DATOS PROPUESTOS

3.1.3 REQUERIMIENTOS DE DESARROLLO

Documentación

Esta sección establece los documentos que serán llevados a cabo a lo largo del proyecto, para poder tener un respaldo documentado de cada uno de los pasos que se llevan a cabo para poder implementar el sistema de información:

- a) Perfil.
- b) Anteproyecto.
- c) Documento Final.
- d) Manual de usuario.
- e) Manual de administración.

Recurso Humano

Es esta sección se definen los requerimientos necesarios para la elaboración del sistema. Para el desarrollo del sistema de información se requiere el trabajo de tres analistas programadores en cada una de las siguientes etapas:

- a) Estudio de la situación actual.
- b) Planeación.
- c) Análisis y definición de requerimientos.
- d) Estudio de la factibilidad.
- e) Diseño.
- f) Programación y prueba.
- g) Documentación.

El equipo de trabajo debe contar con las siguientes características:

- a) Capacidad de trabajar en equipo.
- b) Responsabilidad.
- c) Capacidad de investigación y análisis.
- d) Experiencia en programación orientada a objetos.
- e) Conocimiento en desarrollo de aplicaciones cliente- servidor.
- f) Conocimiento de administración de base de datos en SQL Server 2008 R2 Express.
- g) Capacidad de desarrollo de aplicaciones con VB .NET 2010.

Plataforma.

La plataforma .NET de Microsoft está diseñada para que se puedan desarrollar componentes software utilizando casi cualquier lenguaje de programación, de forma que lo que se escriba en un lenguaje pueda utilizarse desde cualquier otro de la manera más transparente posible. Permitiendo no limitarse a un único lenguaje de programación, permitiendo cualquier lenguaje de programación, que se adhiera a normas comunes establecidas para la plataforma .NET en su conjunto. Por lo que es necesario utilizar el Framework 4.0 para el desarrollo de este proyecto, tanto en las estaciones de trabajo y en los equipos de desarrollo.

Lenguaje de programación.

Este se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos, que siguen los flujos de la información de la realidad que se intenta abstraer.

Por lo que al realizar un sondeo de las posibles aplicaciones a utilizar se pudo recopilar en la siguiente matriz:

CARACTERÍSTICA	VS.NET	JAVA	PHP
Minimiza tiempo para el desarrollo de aplicaciones	X		
Entorno de desarrollo integrado	X	X	
seguridad en código fuente y aplicación	X	X	
Soporte en el mercado nacional	X	X	
Rendimiento	X	X	X
Integración con el gestor de BD	X		X
Integración con el servidor Web	X	X	X
Integración con el sistema operativo	X	X	
Desarrollo con propósitos generales	X	X	
Escalabilidad	X	X	X
Estabilidad	X	X	X

Tabla 9: Características de lenguajes de programación

3.1.4 REQUERIMIENTOS ECONÓMICOS

Para poder implementar este proyecto es necesario invertir en hardware y software, ya que actualmente el cementerio municipal santa Isabel cuenta con pocos recursos para implementar el proyecto, por lo que se ha solicitado se incluya en el presupuesto del año 2011 para poder implementar este proyecto.

No.	RUBRO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
1.-	ESTACIONES DE TRABAJO	2	\$ 848.00	\$ 1,696.00
2.-	UPS (700 VA)	3	\$ 46.00	\$ 138.00
3.-	IMPRESOR FX-890	2	\$ 450.00	\$ 900.00
4.-	SwitchLinksys de 8 puertos	1	\$ 29.99	\$ 29.99
			TOTAL	\$ 2,763.99

Tabla 10: Equipo solicitado (A partir del anexo 7.4.2 para más detalles)

3.2 SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN Y GESTOR DE BASES DE DATOS PARA EL DESARROLLO DEL SISTEMA

Para poder determinar las herramientas necesarias para diseñar, desarrollar e implementar el sistema de información propuesto para el Cementerio Municipal Santa Isabel se analizó el tipo de escenario que presenta actualmente el cementerio y la alcaldía municipal de Santa Ana.

Actualmente el cementerio solamente cuenta con una computadora para realizar sus procesos digitalmente y no cuentan con una red local, por lo que para implementar el sistema es necesario generar un escenario que reúna las condiciones mínimas necesarias para la correcta operación del mismo.

El departamento de informática de esta institución posee las licencias de desarrollo de Microsoft Visual Studio 2010 como plataforma de desarrollo de sus aplicaciones de escritorio, por lo que es conveniente la selección de las siguientes herramientas para desarrollar el sistema de información:

1. Visual Basic .NET 2010
2. SQL Server 2008 R2 Express

Esta selección es la adecuada debido a que ellos serán los encargados de brindar el mantenimiento del sistema y no se infringirá ninguna ley de derechos de autor. Así mismo estas tecnologías cuentan con suficiente soporte para el desarrollo, por lo que esto permitirá que el proyecto se desarrolle en un menor tiempo, dando un mejor resultado cuando sea implementado.

3.3 ESTUDIO DE FACTIBILIDAD

3.3.1 FACTIBILIDAD TÉCNICA

a.- Selección de Hardware y Software

Actualmente el Cementerio Municipal Santa Isabel, solamente cuenta con un equipo de informática como ya se mencionó anteriormente. Sin embargo la alcaldía municipal se encuentra en disposición de compra del equipo necesario para la implementación de este proyecto, debido a que reconoce la importancia del servicio para la comunidad santaneca. La selección del equipo solicitado se ha realizado en base a las condiciones económicas con que cuenta la institución y la tecnología existente en el mercado actualmente. Logrando una implementación con los equipos necesarios en una inversión sostenible para la Alcaldía Municipal de Santa Ana. Así mismo se ha tomado en cuenta la escalabilidad del sistema para la selección de los equipos con las características siguientes que se describen en las tablas 11, 12 y 13:

1.- Estaciones de trabajo

DELL OPTIPLEX 380		
No	COMPONENTES	DESCRIPCION
1.-	PROCESADOR	Intel® Pentium® E5400 (2MB Caché, 2.70GHz, 800MHz FSB)
2.-	SISTEMA OPERATIVO	Windows® 7 Profesional Original de 32 bit con Medio en Español
3.-	MEMORIA	Memoria de 2GB Dual Channel DDR3 SDRAM 1333MHz - 2DIMMs
4.-	MONITOR	Monitor Dell E1910H - Plano y Pantalla Ancha de 18.5 pulgadas

5.-	TARJETA DE VIDEO	Video Integrado, Intel® GMA X4500
6.-	DISCO DURO	Disco Duro de 250GB Serial ATA (7200RPM) con DataBurst Caché™
7.-	DISPOSITIVO ÓPTICO	Single Drive: 16X (DVD+/-RW) Burner Drive
8.-	TECLADO DELL	Dell Multimedia Teclado – Español
9.-	MOUSE DELL	Dell Laser Mouse - Black Gloss
10.-	LECTOR DE MEMORIA	Lector de Tarjeta de Medios Dell 8-in-1
11.-	BOCINAS DELL	Bocinas Dell AX210 Estéreo USB
12.-	MODEM	56K PCI Data Fax Modem
13.-	TARJETA DE RED	Integrado PCIE 10/100/1000
14.-	GARANTÍA	Un Año de garantía limitada. Servicio en el sitio con respuesta el siguiente día laborable.

Tabla 11: Características de equipo

2.- Otros equipos

No	COMPONENTES	DESCRIPCION
1.-	IMPRESOR FX-890	<ul style="list-style-type: none"> • Tecnología de Impresión Matricial de 9 agujas. • Draft alta velocidad, 12 caracteres por pulgada 680 caracteres por segundo. • Carro: 10 pulgadas. • Memoria BUFER: 128 Kb. • Impresión con copias: Una original + 5 copias. • Ancho máximo de línea de impresión: 80 columnas a 10 caracteres por pulgada / 136 columnas a 10 caracteres por pulgada. • Papel: Hojas sueltas (manual): Anchura: De 10 a 25.7 cm, Longitud: De 10 a 36.4 cm, Grosor: De 0,065 a 0,14 mm, Gramaje: 52,3 a 90 g/m2, Papel continuo: Anchura: De 10.1 a 25.4 cm, Longitud: Hasta 55.9 cm, Grosor: De 0,065 a 0,46 mm, Gramaje: 52,3 a 82 g/m2, 40 a 58 g/m2 (impreso con copias) (válido para etiquetas sobre papel continuo) Número de copias:

		<p>Original + 5 copias, Sobres Tamaño Número 6 16.6 x 9.2 cm, Número 10 24 x 10.4 cm, Grosor: De 0,16 a 0,52 mm, Gramaje: De 45 a 90 g/m2.</p> <ul style="list-style-type: none"> • Puertos: <ul style="list-style-type: none"> ○ Paralelo ○ USB. • Sistemas operativos soportados: <ul style="list-style-type: none"> ○ Windows 95, 98, 2000, Me, XP, Vista y 7 • Cinta: Cartucho de cinta negra (C13S015329).
2.-	UPS (750 VA)	APC Smart-UPS, 500 Watts / 750 VA
3.-	SwitchLinksys	8 puertos, 10/100Mbps

Tabla 12: Otro equipo

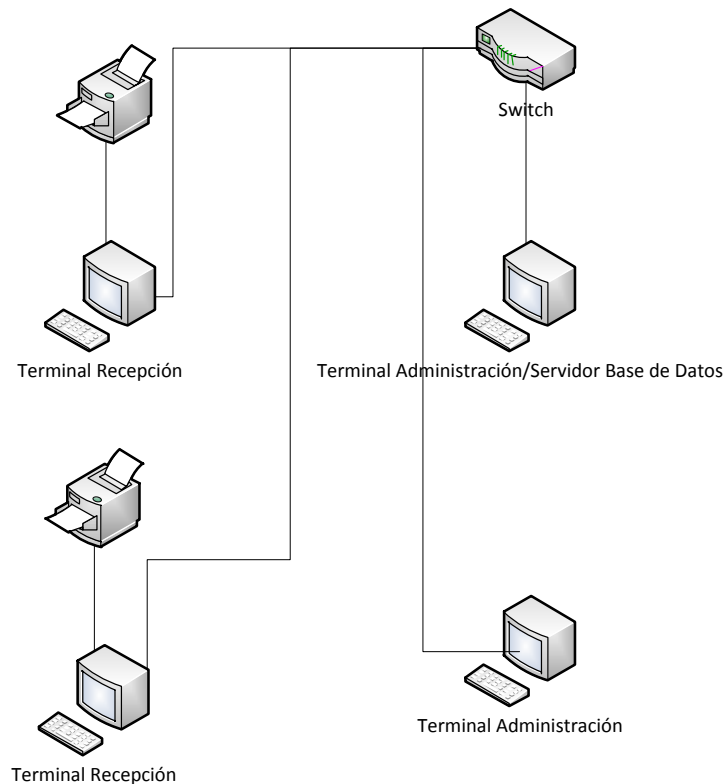
3.- Software

No	COMPONENTES	DESTINO
1.-	Microsoft Windows 7 Profesional-Español	Estaciones de trabajo
2.-	Microsoft Office Home and Business 2010 - Español	Estaciones de trabajo
3.-	SQLServer 2008 R2 Express	Estación de trabajo (Servidor)
4.-	Trend Micro Worry-Free Business Security Services	Estaciones de trabajo

Tabla 13: Software

b.- Distribución de equipos

La distribución del equipo informático solicitado para este proyecto se hará en base a las necesidades que presenta el Cementerio Municipal Santa Isabel, según se pudo constatar en las entrevistas al personal, junto al análisis de la situación actual. Según se muestra en la siguiente figura:



c.- Capacidad de mantenimiento de equipos

Para este proyecto se cuenta con el apoyo del Departamento de Tecnología de la Información de la Alcaldía Municipal de Santa Ana, ya que ellos serán los encargados de brindarles el mantenimiento necesario a los equipos que se utilicen en el proyecto al ser puesto en marcha. Así mismo serán los encargados de brindar el soporte a los usuarios del Cementerio Municipal Santa Isabel después de implementado el proyecto, por lo que se les entregara el paquete de documentación del proyecto para que puedan desempeñar esta tarea de manera más eficiente, distribuyéndose de la siguiente manera:

1. Manual de usuario del Sistema de Información.
2. Manual de administración del Sistema de Información.

Por todo lo anterior podemos concluir que este proyecto es tecnológicamente factible, ya que actualmente la tecnología propuesta para el desarrollo de la solución de la problemática del Cementerio Municipal Santa Isabel, se encuentra disponible en el mercado y facilitando las condiciones de desarrollo, la Alcaldía Municipal de Santa Ana cuenta con la licencia de desarrollo de la plataforma de programación. Por lo que no se

cuentan con problemas de licenciamiento y la infracción a los derechos de autor para poder implementar el proyecto.

Así mismo no se cuenta con resistencia al cambio por parte de los trabajadores del cementerio, ya que ellos al ser entrevistados consideran que la institución se encuentra desfasada tecnológicamente en la manera de atender al público y en sus procesos internos, por lo que ven en el desarrollo de este proyecto la posibilidad de mejorar el servicio a los usuarios, así como facilitar su trabajo haciéndolo más eficaz y eficiente. Lo que facilita la inducción para la utilización del sistema, ya que los usuarios dispondrán de un mejor nivel de aprovechamiento de la misma.

3.3.2 FACTIBILIDAD ECONÓMICA

Con el Diseño y Desarrollo del Sistema de Información se busca obtener un mejor desempeño de los procesos internos de la institución así como también una reducción de tiempo de respuesta a los usuarios para brindar los servicios que esta institución presta a la comunidad. Por lo que se presenta el estudio que dio como resultado la factibilidad económica del proyecto, que nos permite hacer una comparación entre la relación de los costos de la implementación del sistema de información y los beneficios que percibirá el cementerio municipal santa Isabel.

Con la implementación de este sistema se pretenden lograr beneficios que contribuyan a un mejor funcionamiento del cementerio:

Beneficios tangibles

Los beneficios tangibles aportados por el sistema propuestos están dados por los siguientes aspectos:

- Reducción de horas de trabajo extra, que los empleados actualmente deben realizar en temporadas específicas del año, reduciendo los días compensatorios.
- Mejora de la calidad del servicio prestado y los controles de la recepción de aranceles percibidos.
- Reducción de los tiempos de atención a los usuarios del servicio que presta el cementerio.
- Control y seguimiento de las tareas del plan de trabajo que permitirá un mejor y más efectivo empleo de los recursos.

Beneficios intangibles

Los beneficios intangibles aportados por el sistema propuestos están dados por los siguientes aspectos:

- Modernización y optimización de los procesos administrativos.
- Generar información más eficiente y confiable, que sirva para la toma de decisiones oportunas.
- Mejor capacidad de búsqueda y actualización de la información, reduciendo la fuerza de trabajo en la realización de los procesos y el control de los recursos, facilitando las labores del personal.
- Capacidad de almacenar y proteger datos de manera estandarizada para generar información, que permita la persistencia de la misma a través del tiempo.

PRESUPUESTO DE HARDWARE Y SOFTWARE				
No.	RUBRO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
1.-	ESTACIONES DE TRABAJO	2	\$ 848.00	\$ 1,696.00
2.-	CABLEADO ESTRUCTURADO	*	\$ 278.00	\$ 278.00
3.-	UPS (700 VA)	3	\$ 112.50	\$ 450.00
4.-	IMPRESOR FX-890	2	\$ 450.00	\$ 900.00
5.-	POLARIZACION	5	\$ 62.30	\$ 311.50
6.-	SWITCHLINKSYS	1	\$ 29.99	\$ 29.99
TOTAL				\$3,665.49

TABLA 14: Presupuesto de Hardware y Software (A partir del anexo 7.4.2 para más detalles)

En la tabla 16 se detalla el costo que tendrá el sistema durante todo su periodo de operación, el cual se ha estimado en una vida útil de 5 años, dicho costo será el de energía eléctrica. Cabe mencionar también que a parte del costo de energía eléctrica también se consideran el del desarrollo mismo al inicio del proyecto.

A continuación se detalla el procedimiento para obtener el consumo de energía durante una jornada de 10 horas, cabe aclarar que el equipo de cómputo se utilizara durante todo el mes, ya que debido al tipo de servicios que la institución presta no es posible suspender las operaciones durante cierto día a lo largo del año.

Otro aspecto que es necesario aclarar es con respecto al mantenimiento del equipo de cómputo, ya que este no se ha incluido como costo fijo debido a que este será realizado por parte del departamento de informática de la Alcaldía Municipal.

Inversión del proyecto: \$3,215.49	Costo de consumo de energía eléctrica del equipo de cómputo al año: \$365.28, proyectado para cuatro años es: \$1461,12
	Consumo de watts del servidor 300 W
	Convirtiendo Watts a KW KW = 1W / 1000 KW = 300 W / 1000 KW = 0.3
	Consumo de KW del servidor 300 W = 0.3 KW
	Costo de KW/h en El Salvador \$0.1691
	Horas de funcionamiento del servidor al día 10 h
	Horas de funcionamiento del servidor al mes 300 h
	Consumo de KW del servidor al mes: 0.3 KW * 300 h = 90 KW/h
	Costo estimado de consumo al mes de KW del servidor 90 KW/h * \$0.1691 \$15.22
	Costo estimado de consumo al año de KW del servidor \$15,22 * 12 = \$182,64
Costo de energía por 3 equipos \$182.64*2 = 365.28	
\$3,215.49	\$365.28

TABLA 15: Costos del Proyecto y Costo de Operación

Análisis Costo / Beneficio

En la tabla 16 se muestra el análisis costo / beneficio con respecto al sistema de información, el cual permitirá comprobar si el proyecto es factible o no.

Bien, para este análisis es necesario identificar primeramente los beneficios que este proporcionara una vez implementado y los costos que este requiere para la elaboración y funcionamiento del mismo.

Como se mencionó anteriormente, los únicos costos que se perciben son en la fase de desarrollo y en el pago de energía eléctrica el cual se considera costo para por parte de los equipos de cómputo.

Una vez identificados los costos y los beneficios se procede a realizar el análisis del mismo, mostrando que el sistema desde el punto de vista económico es viable ya que se podrá recuperar la inversión en el mismo año de la implementación según se muestra en la gráfica.

Con respecto a los costos que se estiman fijos durante toda la vida del proyecto se puede decir que estos son relativamente bajos en comparación con los beneficios que este proveerá, ya que estos últimos van en incremento durante el transcurso de los años en contraposición a los gastos que este demanda para su funcionamiento.

Por lo cual se puede decir que el proyecto desde el punto de vista económico es viable ya que las ganancias o beneficios que este generara son mucho mayores a los costos de operación del mismo.

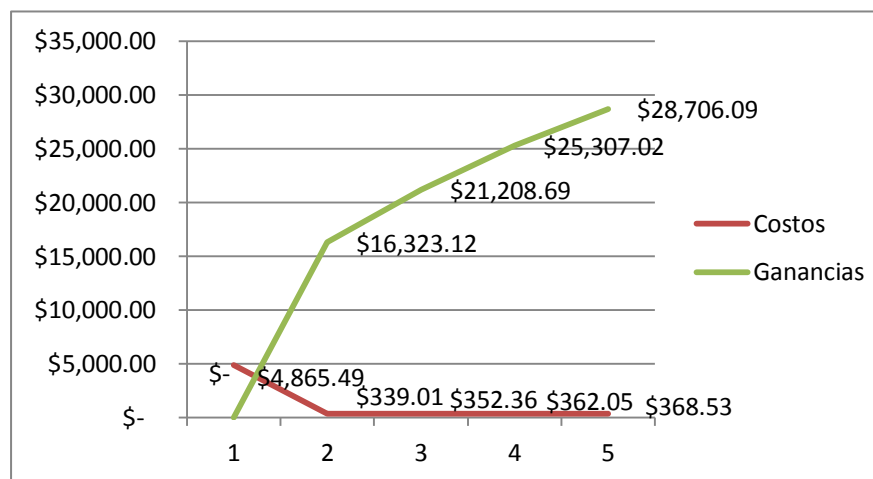
RUBROS	Año de Desarrollo		Años de vida útil			
	Año 0	1	2	3	4	TOTAL
Costo	\$ 3,215.49	\$ 365.28	\$ 409.09	\$ 452.92	\$ 496.75	
Factor de descuento	1.00	0.93	0.86	0.80	0.74	
Costo con descuento	\$ 3,215.49	\$ 339.01	\$ 352.36	\$ 362.05	\$ 368.53	\$ 4,637.43
Beneficio	\$ -	\$ 17,588.16	\$ 24,623.42	\$ 31,658.69	\$ 38,693.95	
Factor de descuento	1.00	0.93	0.86	0.80	0.74	
Beneficio con descuento	\$ -	\$ 16,323.12	\$ 21,208.69	\$ 25,307.02	\$ 28,706.09	\$ 91,544.92
Beneficio Real – Costo Real	\$ (4,865.49)	\$ 15,984.11	\$ 20,856.33	\$ 24,944.97	\$ 28,337.56	
Beneficio Acumulado - Costo	\$ (4,865.49)	\$ 11,118.62	\$ 31,974.96	\$ 56,919.93	\$ 85,257.49	

TAZA DESCUENTO

7.75%

ARANCELES	COSTO
Mantenimiento y vigilancia	\$ 6.00
Enterramiento	\$ 14.42
Compra de fosa	\$ 101.72
TOTAL	\$ 122.14

Cant. Fallecidos Promedio Mínimo (Anuales)	144
Beneficio Anual	\$ 13,191.12
Beneficio de desarrollo	\$ 2,700.00
Beneficio Total	\$ 15,891.12



NPV = Total de Beneficios – Total de Costos

NPV = \$ 86,907.49

TABLA 16: Análisis de factibilidad económica

De esta forma se prevé que aumente la productividad para desarrollar los diferentes procesos, haciendo así que los empleados realicen las labores de manera más ágil, dando como resultado a cada proceso menos horas hombre, reduciendo los días compensatorios de los empleados por la colaboración de horas extras para suplir la demanda en determinadas temporadas altas del año.

Así mismo, esta factibilidad es sustentada con la gestión del presupuesto asignado al cementerio por la alcaldía Municipal de Santa Ana, necesario para la implementación de este proyecto. Que es acorde a la solicitud del desarrollo de este proyecto por parte de la administración.

CAPÍTULO IV.- DISEÑO Y DESARROLLO DEL SISTEMA DE INFORMACIÓN

4.1 ESTÁNDARES PARA ARCHIVOS, CAMPOS Y VARIABLES

Para el desarrollo de la aplicación se han utilizado diferentes tipos de estándares en la metodología de trabajo, a fin de facilitar el desarrollo acortando su tiempo y logrando un mejor entendimiento del código fuente, así como amparándose en las ventajas que ofrece la programación orientada a objetos.

Nombramiento de variables.

El nombramiento de variables a través del estándar camelcase (PrimeraPalabra), es decir que toda variable utilizada deberá tener la letra inicial de cada una de las palabras por la que este computa con letra mayúscula. Este estándar fue adoptado debido a la posible necesidad de nombrar variables con compuestas de más de una palabra, por lo que a través de este estándar se puede dar nombres que permitan entender lo que una variable puede contener.

Estándares de modelado de datos

- ✓ El modelado de los datos se desarrollara utilizando una combinación de Lenguaje Unificado de Modelado (UML) y Entidad Relación.
- ✓ Se utilizará para UML únicamente los diagramas de casos de uso y diagrama de clases (para los casos de uso hacer la descripción de los casos de uso)
- ✓ Para el modelo entidad relación se desarrollara el diagrama en herramienta case (PowerDesigner) luego se exportara a SQL Server 2008.

Estándares de programación

- ✓ Se utilizara lenguaje de programación de cuarta generación orientado a objetos (Visual Basic .NET 2010) con un manejador de bases de datos que soporta procedimientos almacenados (SQL Server 2008 R2).
- ✓ La programación estará basada en una arquitectura cliente servidor con un modelo de tres capas (capa de datos, capa de negocio y capa de aplicación) y debe contar con código bien comentáramos el cada una de las capas así como un diccionario de datos bien definido.
- ✓ No usarán asistentes brindado por la herramienta .NET 2010, salvo el caso donde la herramienta no permita hacerlo a través del código.

Estándares de definición de tablas

Las tablas se crearán bajo un esquema distinto al DBO el cual debe hacer referencia al departamento en que se trabajará, según detalle (6 caracteres):

Nombre del Modulo	Nombre del Esquema
Módulo de Usuarios	MODUSU
Módulo de Personal	MODPER
Módulo de Registro de Información	MODRIN
Módulo de Registro de Cobros y Costos	MODCYC
Módulo de Agenda de Actividades	MODAGA

Al seleccionar los esquemas de trabajo se utilizarán tres categorías. Tablas maestras que serán aquellas donde se guardaran los registros que darán vida al sistema propuesto, las tablas catalogo que se encargaran de almacenar todos los catalogo de servicios, roles, etc.; que se utilizarán y las tablas transaccionales que son aquellas que permitirán almacenar las transacciones que se realicen con las tablas maestras y catálogos, obteniendo los siguientes nombres:

✓ **Tablas maestras**

ESQUEMA.MST+NOMBRE(MODPER.MSTEMPLEADOS)

✓ **Tablas catálogos**

ESQUEMA.CAT+NOMBRE(MODPER.CATTIPOEMPLEDOS)

✓ **Tablas transaccionales**

ESQUEMA.TBL+NOMBRE(MODRIN.TBLINGRESOS)

Estándares de definición de Procedimientos Almacenados y Triggers

Los procedimientos almacenados se crearán como base fundamental de la interacción con la base de datos, ya que estos brindan portabilidad, seguridad y rapidez de la obtención de datos al sistema de información. Utilizando la siguiente forma normalizadora:

Convención	Ejemplo
SP_NOMBREPROCEDIMIENTO	SP_GUARDAREMPLEADO
TG_ACCION+TABLA	TG_TRANSACCIONMSTINHUMACION

Estándares de Objetos

Los nombres de los objetos que se utilizarán durante la elaboración de este proyecto deben basarse en las siguientes convenciones:

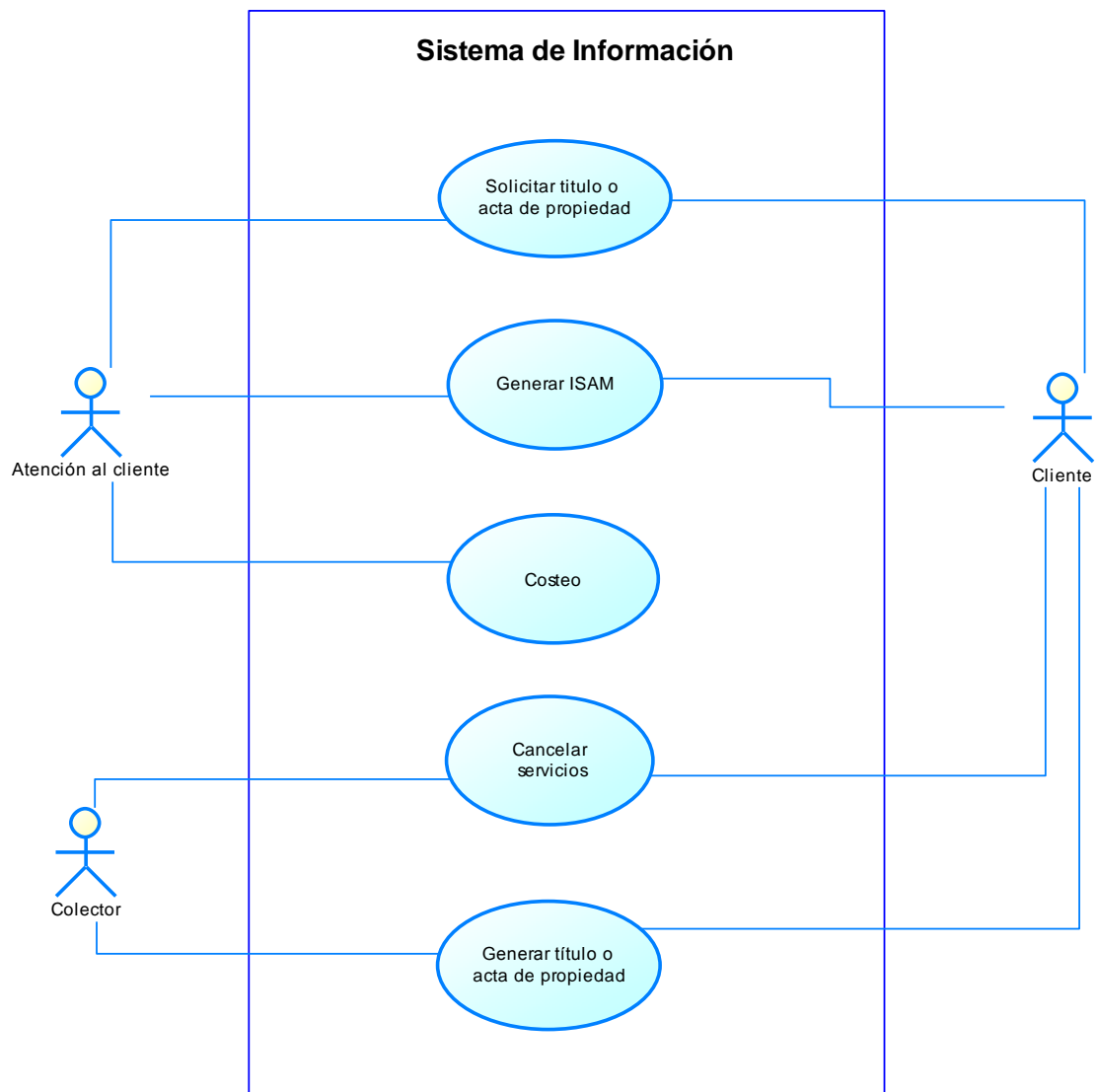
Objeto	Convención	Ejemplo
Formulario	Frm+Nombre	FrmEmpleados
Botones	Btn+Nombre	BtnGuardar
Cajas de texto	Txt+Nombre	TxtDui
Listas(Combobox)	Cmb+Nombre	CmbDepartamentos
Datetimepicker	Dtp+Nombre	DtpFechaSam
Checkbox	Chk+Nombre	ChkEstado
Radio botón	Rbtn+Nombre	RbtnGenero
Datagridview	Dg+Nombre	DgEmpleados
Statusbar	Stb+Nombre	StbEstado
GroupBox	Gb+Nombre	GbOpciones
Dataset	Ds+Nombretabla	DsEmpleados
MenuStrip	Mn+Nombre	MnPrincipal
Clases	Cl+NombreClase	ClInhumacion

Las Propiedades deben definirse con un nombre igual al nombre del campo que está en la base de datos. Así mismo todas las rutinas deben de estar dentro de un “Try Catch”, para el manejo de excepciones dentro de la aplicación a desarrollar.

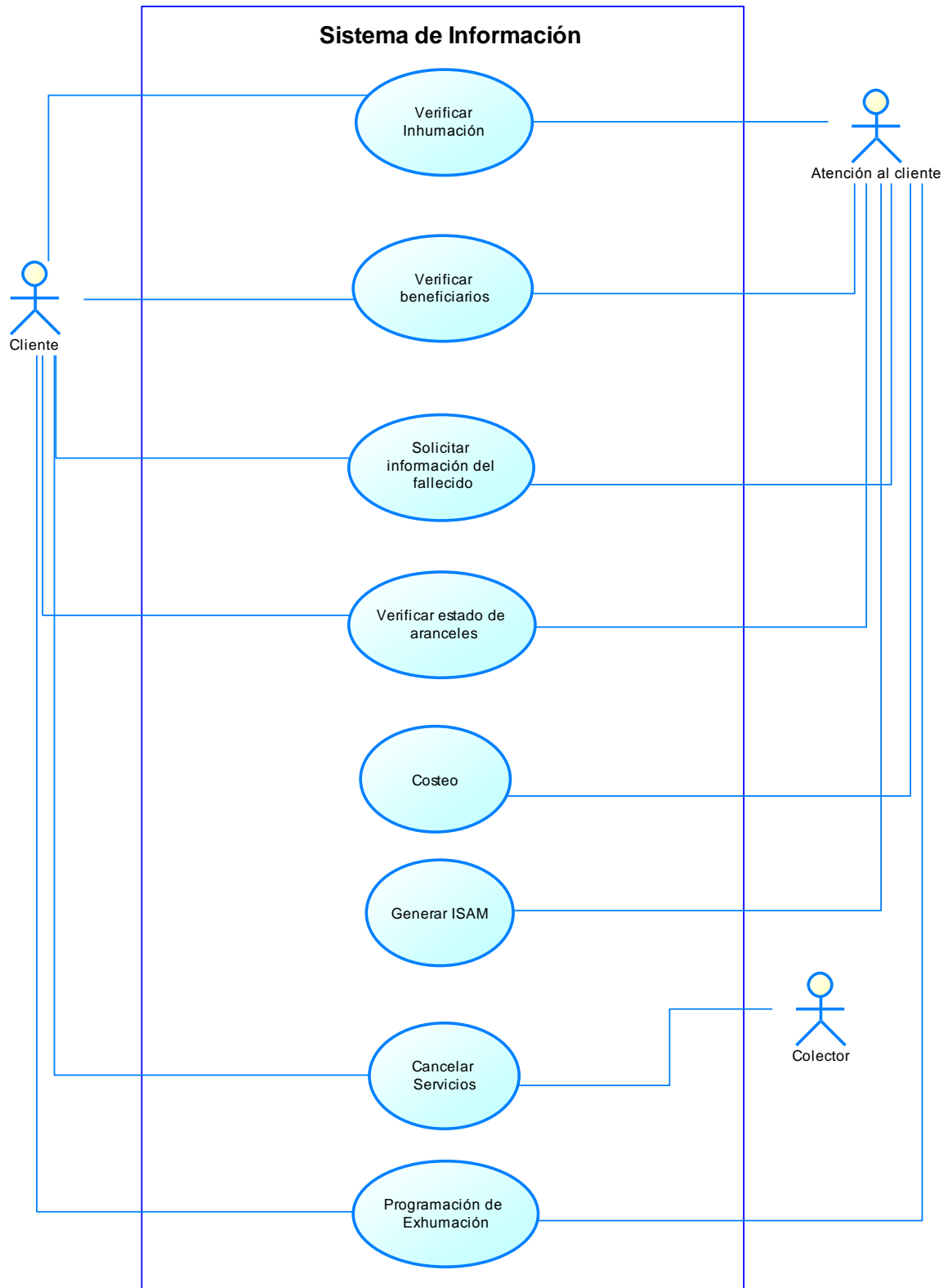
4.2 SISTEMA DE INFORMACIÓN

4.2.1 DIAGRAMA DE CASOS DE USO

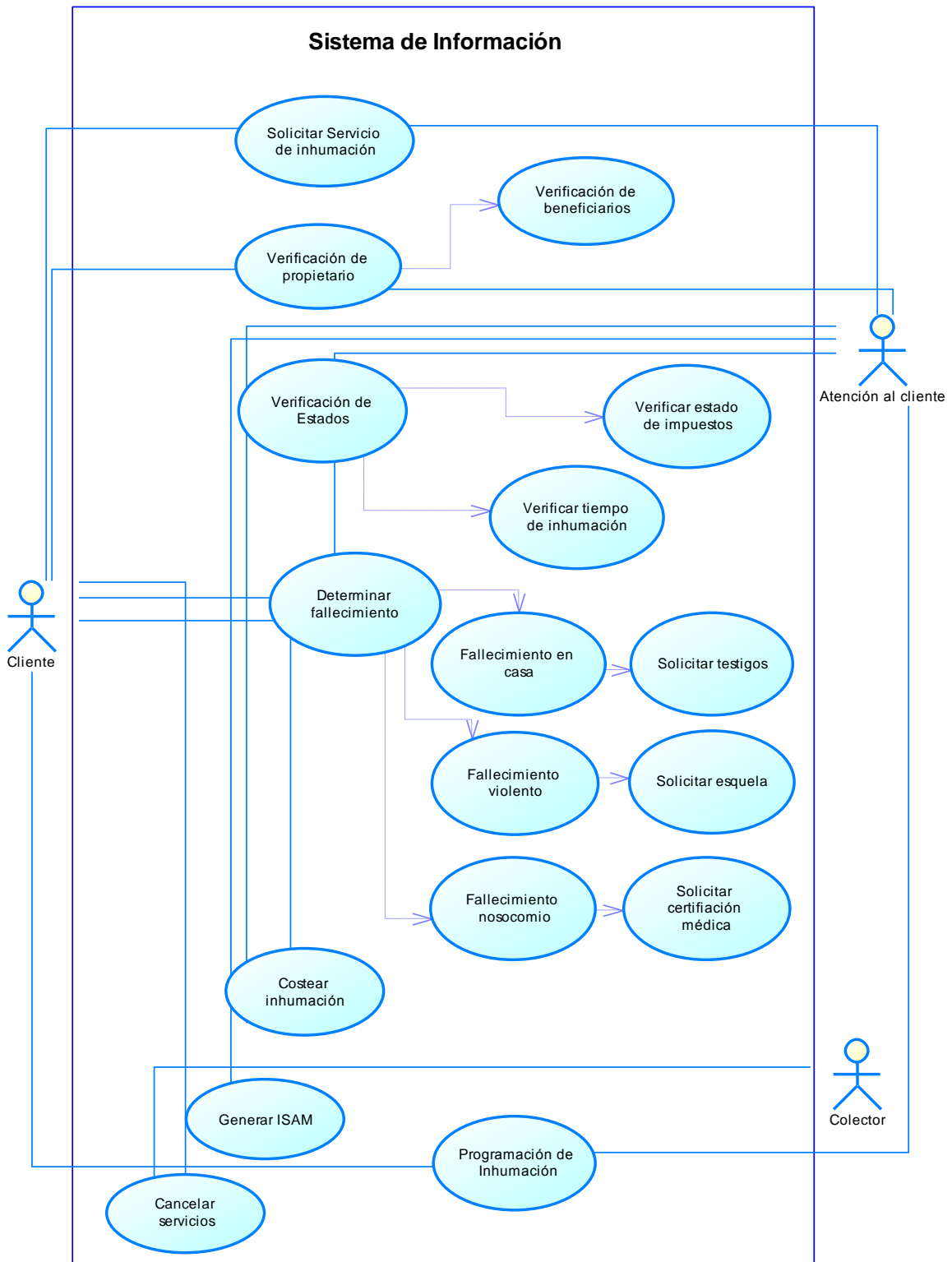
PROCEDIMIENTO PARA GENERAR TITULOS O ACTAS DE PROPIEDAD



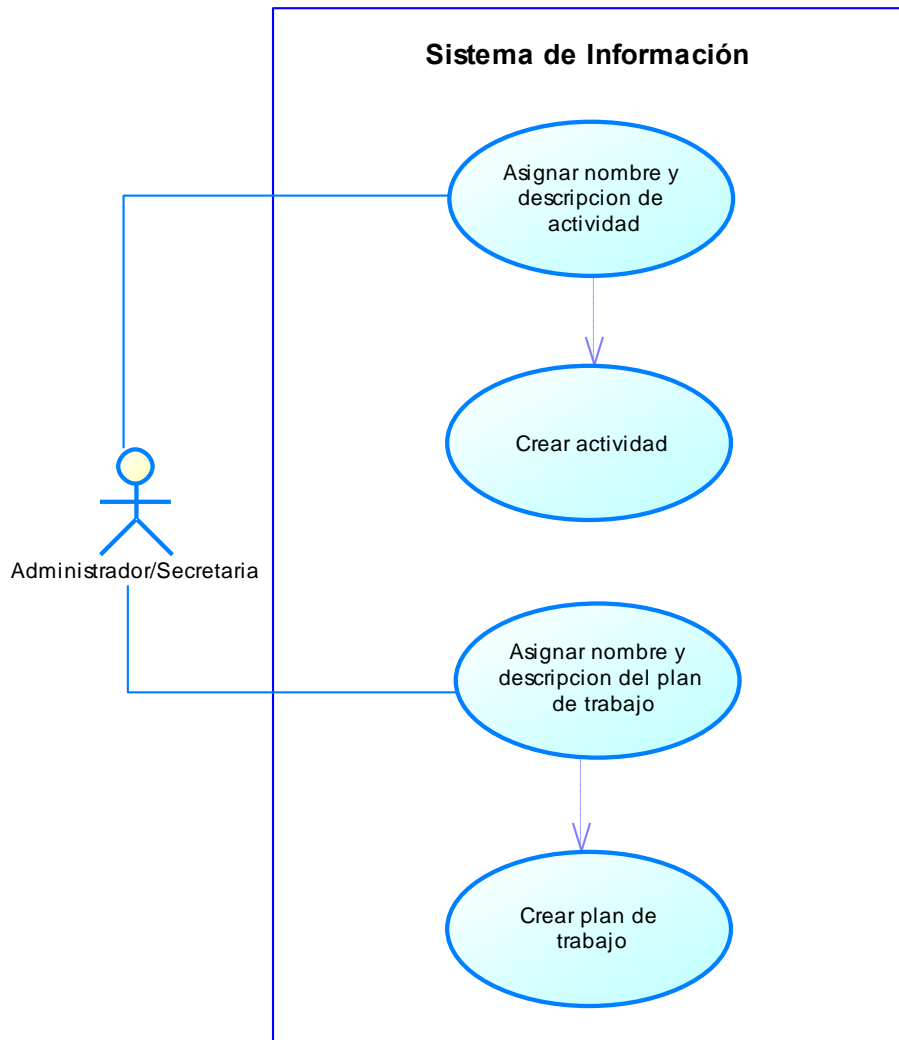
PROCEDIMIENTO DE EXHUMACIÓN



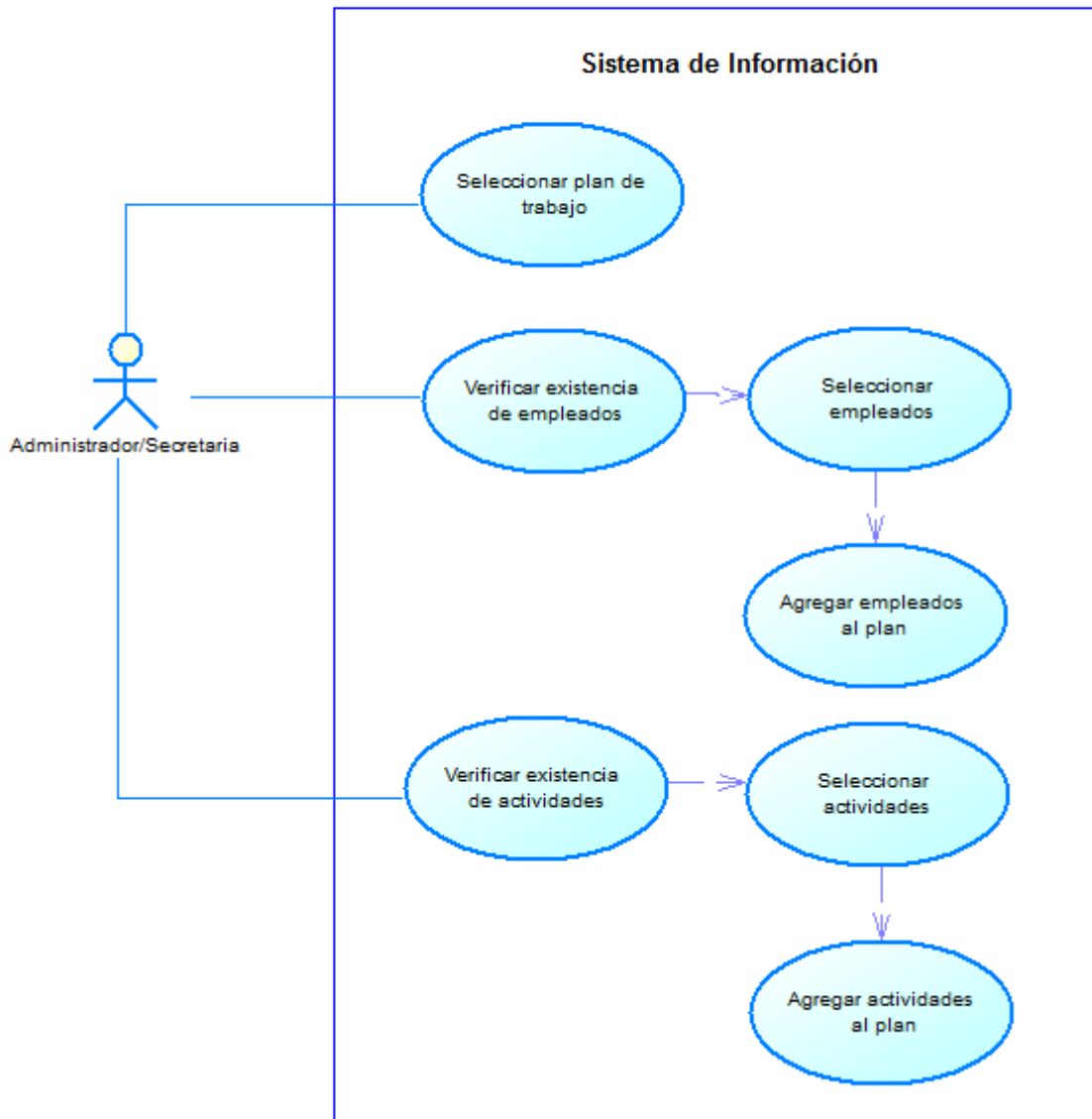
PROCEDIMIENTO DE INHUMACIÓN



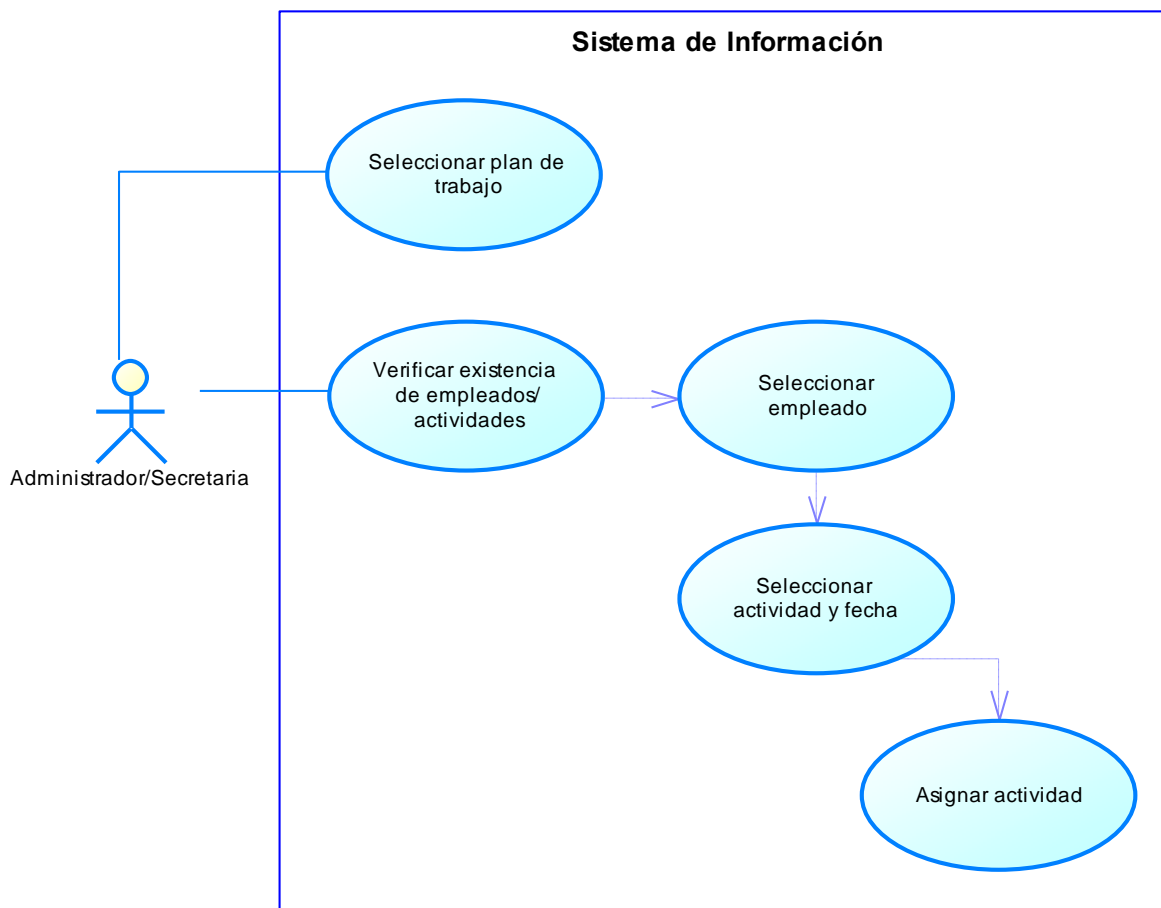
PROCEDIMIENTO PARA CREAR PLANES DE TRABAJO/ACTIVIDADES



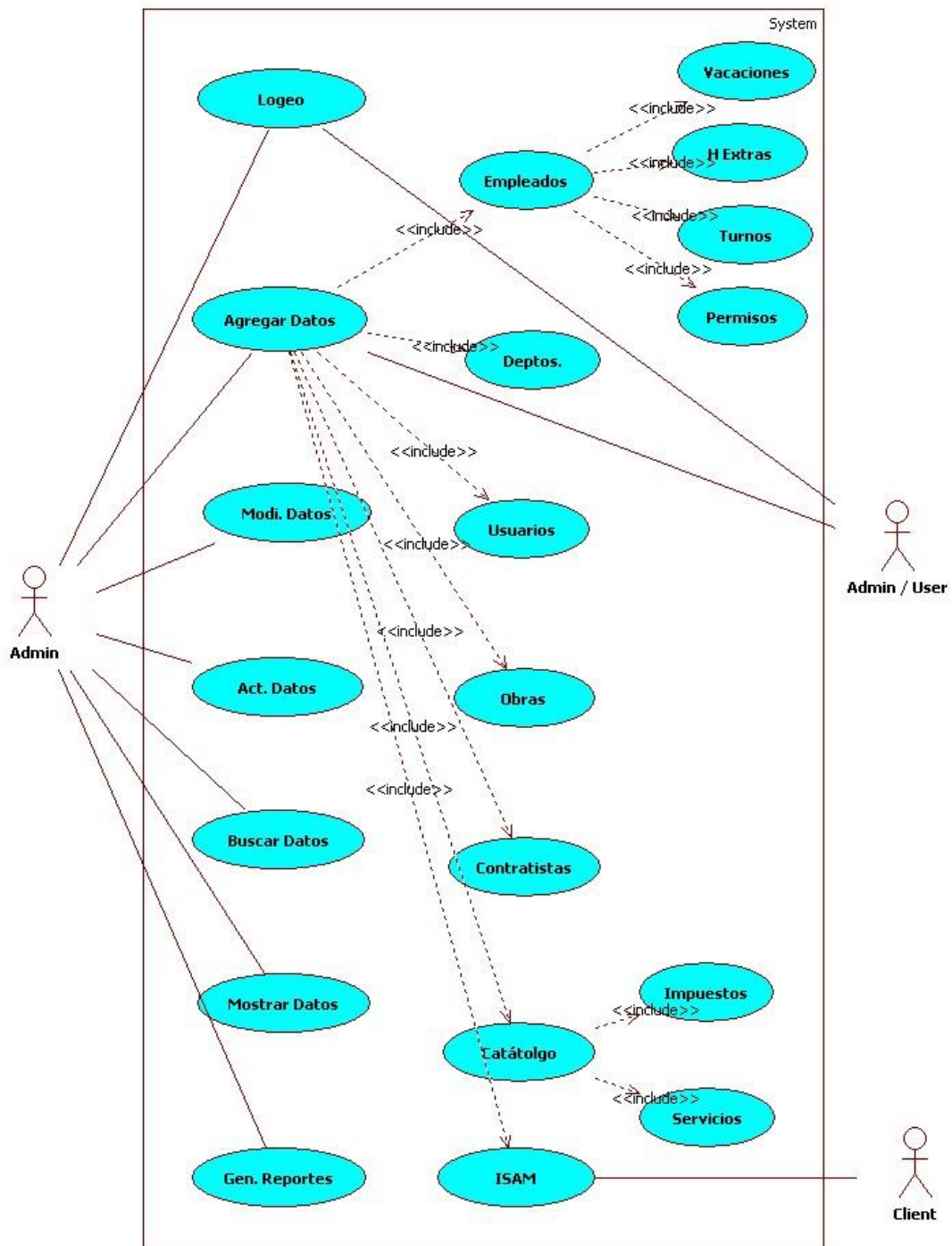
PROCEDIMIENTO PARA AGREGAR ACTIVIDADES/EMPLEADOS AL PLAN DE TRABAJO



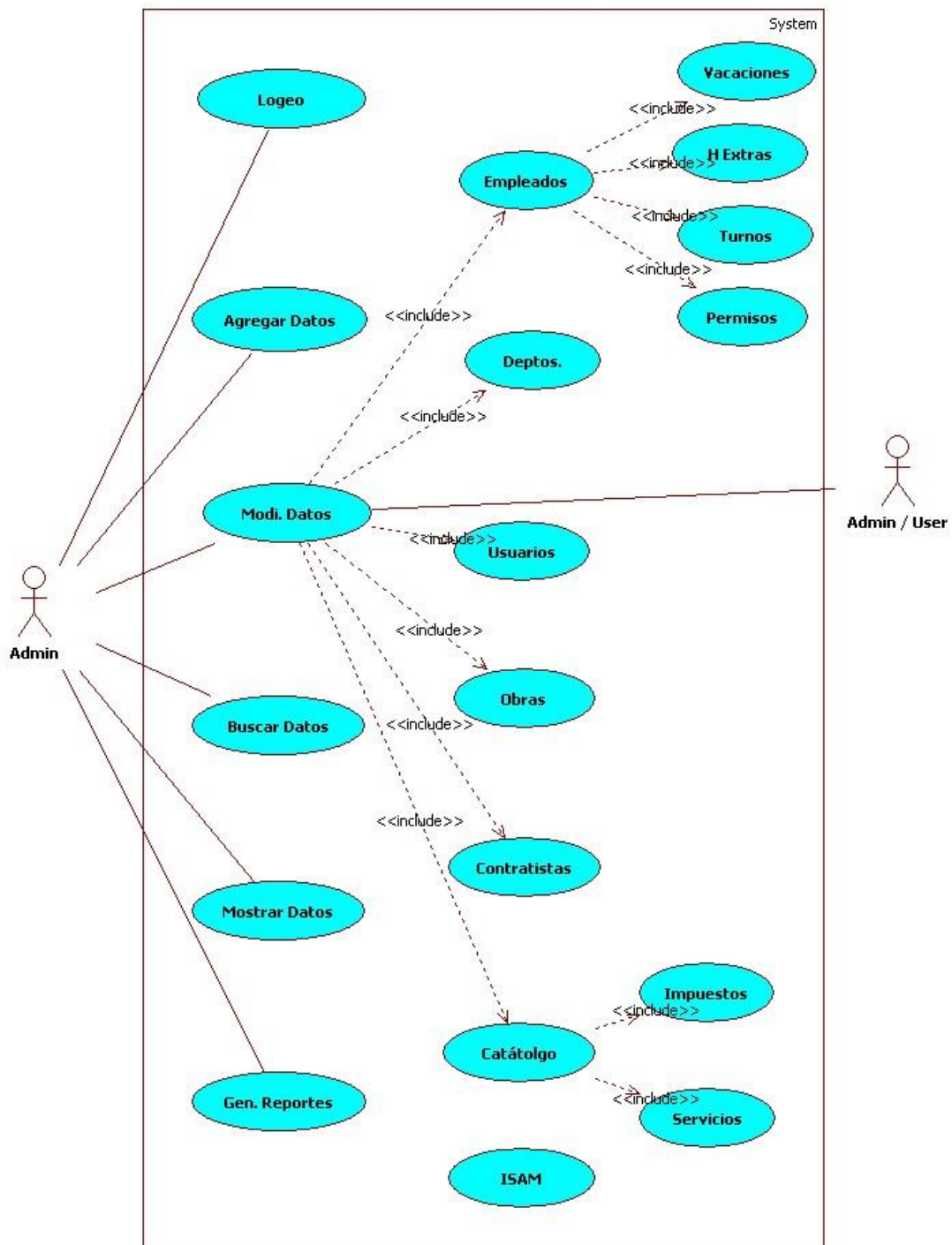
PROCEDIMIENTO PARA ASIGNAR ACTIVIDADES A LOS EMPLEADOS



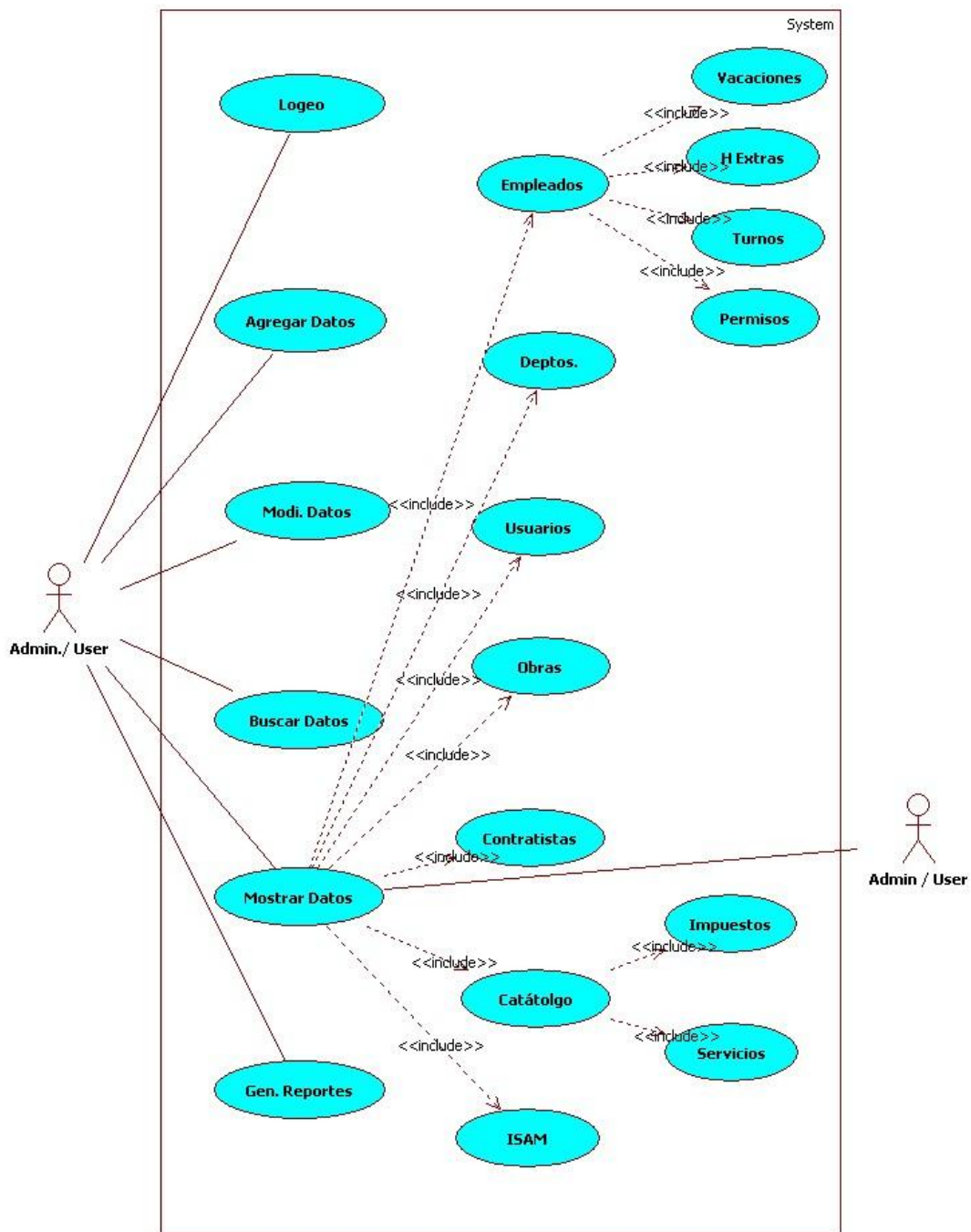
PROCEDIMIENTO PARA AGREGAR DEPARTAMENTOS, EMPLEADOS, CONTRATISTAS, IMPUESTOS, SERVICIOS Y FACTURAS ISAM



PROCEDIMIENTO PARA MODIFICAR DEPARTAMENTOS, EMPLEADOS, CONTRATISTAS, IMPUESTOS Y SERVICIOS



PROCEDIMIENTO PARA MOSTRAR DEPARTAMENTOS, EMPLEADOS, CONTRATISTAS, IMPUESTOS Y SERVICIOS



ANALISIS DE CASOS DE USO

Nombre:	Administración de RRHH.
Autor:	Luis Clavel
Fecha:	12/12/2010
Descripción: Muestra el proceso de registro control de actividades del personal que labora en el Cementerio Santa Isabel.	
Actores: <ol style="list-style-type: none">1. Usuario administrador2. Usuarios del sistema.	
Precondiciones: <ol style="list-style-type: none">1. Antes de ingresar un empleado al sistema este debe estar legalmente contratado así previamente asignado a un departamento.2. Antes de hacer uso de un servicio el cliente debe cancelar el costo del servicio así como los impuestos que sean necesarios.	
Flujo Normal: <ol style="list-style-type: none">1. Cuando una persona es contratada se realiza una captura de los datos personales, así como el departamento al que este pertenecerá y las actividades que tendrá a cargo.2. Una vez recolectados los datos personales del empleado, se lleva un control de las operaciones o solicitudes que este realiza como por ejemplo, horas extras, permisos, turnos, vacaciones y actividades programadas.3. Si en algún determinado momento el empleado realiza cambios en su información personal como por ejemplo estado civil, se proceden a actualizar los datos en el registro interno del cementerio.4. Al finalizar un periodo labora se procede a realizar un recuento de las actividades que cada empleado realizo a lo largo del año laboral, como por ejemplo permisos y horas extras.	

Nombre:	Administración de Contratistas y obras
Autor:	Luis Clavel
Fecha:	12/12/2010
Descripción: Muestra el proceso de registro de personas contratistas y el control de las obras que estos realizan.	
Actores:	
<ol style="list-style-type: none"> 1. Usuario Administrador. 2. Usuarios del sistema. 	
Precondiciones:	
<ol style="list-style-type: none"> 1. Antes de ingresar un contratista al sistema este debe estar legalmente contratado y autorizado para poder laborar en el interior del cementerio. 2. Debe existir una necesidad que amerite la contratación de un contratista para realizar una determinada operación. 	
Flujo Normal:	
<ol style="list-style-type: none"> 1. Se identifica una necesidad y se verifica si es posible realizarla con el personal interno. 2. En caso de requerir personal extra o especializado para realizar una tarea específica se procede contratar a una persona para dicha actividad. 3. Se asignan las actividades. 4. Se lleva el control de cada una de las actividades realizadas por personas así como el costo de cada una de ellas. 	

Nombre:	Elaboración de factura ISAM
Autor:	Luis Clavel
Fecha:	12/12/2010
Descripción: Permite registrar todos los pagos de servicios e impuestos en el Cementerio Santa Isabel.	
Actores:	
<ol style="list-style-type: none"> 1. Usuarios 2. Cliente 	
Precondiciones:	
<ol style="list-style-type: none"> 1. Se debe verificar la identidad de la persona y el tipo de trámite que esta realiza. 	
Flujo Normal:	
<ol style="list-style-type: none"> 1. Solicitar un servicio o pago de impuesto. 2. Proporcionar los documentos de identidad. 3. Llenar la factura indicando el nombre del servicio y el costo del mismo. 4. Cancelar el servicio en efectivo. 5. Entregar una copia de la factura al cliente y guardar la original. 	

Nombre:	Crear planes de trabajo y actividades
Autor:	Guillermo Rafael Vásquez Castaneda
Fecha:	12/12/2010
Descripción: Muestra el proceso para crear planes de trabajo y actividades del cementerio.	
Actores:	
<ol style="list-style-type: none"> 1. Usuario administrador. 2. Usuario secretaria. 	
Precondiciones:	
<ol style="list-style-type: none"> 1. Ninguna. 	

Flujo Normal:

1. El usuario agrega un nombre y una descripción para el nuevo plan.
2. Se crea el nuevo plan de trabajo.
3. El usuario agrega un nombre y una descripción para la nueva actividad.
4. Se crear la nueva actividad.

Nombre:	Agregar actividades/empleados al plan de trabajo
Autor:	Guillermo Rafael Vásquez Castaneda
Fecha:	12/12/2010
Descripción: Muestra el proceso de agregar actividades y personas al plan de trabajo	
Actores: <ol style="list-style-type: none">1. Usuario Administrador.2. Usuarios secretaria.	
Precondiciones: <ol style="list-style-type: none">1. Debe de haber un plan de trabajo creado correspondiente al año actual.2. Se necesita que exista al menos un empleado y una actividad para poder ser agregados al plan de trabajo.	
Flujo Normal: <ol style="list-style-type: none">1. Se seleccione el plan de trabajo correspondiente al año actual.2. Se seleccionan las actividades que se desean agregar al plan de trabajo.3. Se agregan las actividades al plan de trabajo.4. Se seleccionan los empleados que se desean agregar al plan de trabajo.5. Se agregan los empleados al plan de trabajo.	

Nombre:	Asignar actividades a los empleados
Autor:	Guillermo Rafael Vásquez Castaneda
Fecha:	12/12/2010
Descripción: Muestra el proceso de asignar actividades para que sean realizadas por los empleados.	
Actores:	
<ol style="list-style-type: none"> 1. Usuario Administrador. 2. Usuarios secretaria. 	
Precondiciones:	
<ol style="list-style-type: none"> 1. Debe de haber un plan de trabajo creado correspondiente al año actual. 2. Se necesita que el plan tenga agregados al menos un empleado y una actividad. 	
Flujo Normal:	
<ol style="list-style-type: none"> 1. Se seleccione el plan de trabajo correspondiente al año actual. 2. Se selecciona el empleado del plan de trabajo. 3. Se selecciona la actividad del plan de trabajo y se le asigna una fecha de inicio de la actividad. 4. Se asigna la actividad al empleado. 	

Nombre:	Registro de inhumaciones
Autor:	Luis Chávez
Fecha:	12/12/2010
Descripción: Permite registrar las diferentes inhumaciones que son ofrecidas por el cementerio municipal Santa Isabel.	
Actores:	
<ol style="list-style-type: none"> 3. Atención al cliente 1. Cliente 2. Colector 	
Precondiciones:	
<ol style="list-style-type: none"> 1. Verificación del dueño a través de un documento de identidad y del título de compra 	

del nicho a perpetuidad, realizando una búsqueda en los registros.

2. Si el dueño del nicho ya falleció, el título de perpetuidad solicita 3 beneficiarios, por lo que cualquiera de estos puede hacer uso del servicio y están obligados a pagar los impuestos. Estos son comprobados con el registro encontrado en la precondition anterior.
3. Si el cliente no ha adquirido con anterioridad el servicio, el encargado de atención al cliente se encarga de llenar el formulario con todos los datos necesarios para adquirir el servicio que se desea, pudiendo ser este en las diferentes modalidades que ofrece el cementerio.

Flujo Normal:

1. Si el cliente ya cuenta con un título de propiedad, se procede a verificar todos los datos en el sistema, verificando su solvencia con la institución.
2. Con esta información se verifica cuantas osamentas se encuentran en su propiedad y se verifica quien es el último enterrado para saber si se encuentra solvente con sus impuestos, si no se encuentra solvente debe cancelar los aranceles necesarios para utilizar el servicio (según el catálogo de precios). Y si el puesto ya no tiene nichos disponibles y se cumple con un tiempo de 7 años desde el último entierro, se puede solicitar una exhumación para poder enterrar en ese nicho, caso contrario no se puede hacer el entierro en ese nicho.
3. Si el fallecimiento es en casa, debe llevar 2 testigos al cementerio con DUI y que puedan firmar para confirmar el suceso, para ser registrado en el formulario
4. Si es por una muerte de violencia, medicina legal entrega una esquela (documento que ampara el reconocimiento del cadáver y avala la causa de la muerte), el cual deberá ser anotado en el formulario de registro de inhumación, para guardar una referencia digital de ese documento.
5. Si la muerte fue en un hospital, debe entregar la documentación que se entrega en el hospital justificando la causa de fallecimiento, para guardar una referencia digital de ese documento.
6. Se programa en el calendario de actividades el día y la hora, según la disponibilidad del personal para realizar la exhumación.
7. Se genera el formato ISAM para cobrar los servicios que se han solicitado y luego cancelarse en caja.
8. Se imprime el formato ISAM para que el cliente pueda cancelar los servicios

requeridos, proporcionando el recibo original al cliente y se guardan las demás copias para trabajo interno de la institución.

Flujo Alternativo:

1. Si el cliente ya contaba con el servicio y no se encuentra solvente con los impuestos y aranceles, entonces se le cargará a su cuenta en la generación del formato ISAM.
2. El sistema hace verificación de los datos a guardar del registro, en caso que no sean correctos o que los campos requeridos estén vacíos envía un mensaje para que los corrija.

Poscondiciones

1. Si se compra un puesto para nicho, el cliente está obligado a construir un mausoleo para poder enterrar en ese espacio. Si el cliente ya tenía comprada una fosa puede optar por pagar un nicho, lo solicita y paga el nuevo puesto para construir sobre ese espacio de tierra para cambiar el tipo de construcción del lugar de entierro.
2. Según el reglamento de la ley general de cementerios, las horas hábiles para inhumaciones serán de las siete a las dieciséis horas. A menos que la CSJ ordene que se haga en algún caso especial este servicio o el MSPAS para evitar propagación de enfermedades infecto-contagiosas.

Nombre:	Registro de exhumaciones
Autor:	Luis Chávez
Fecha:	12/12/2010
Descripción: Permite registrar las exhumaciones que son realizadas por el cementerio municipal Santa Isabel.	
Actores: <ol style="list-style-type: none">1. Atención al cliente2. Cliente3. Colector	
Precondiciones: <ol style="list-style-type: none">1. Esta puede ser solicitada por la Corte Suprema de Justicia para la investigación de sobre seguimiento de algún caso judicial.	

2. Puede ser ejecutada de oficio por el cementerio si los familiares del fallecido no pagan los impuestos de refrenda cada 7 años.

Flujo Normal:

1. Verificación en el sistema del dueño, a través de un documento de identidad y del título de compra del lugar de inhumación.
2. Si el dueño ya falleció, el título posee 3 beneficiarios, por lo que cualquiera de estos puede solicitar los servicios prestados por el cementerio. Los cuáles serán obtenidos a través de la búsqueda del flujo anterior.
3. Se registra la solicitud de exhumación, llenando todos los datos solicitados por el formulario del sistema de información.
4. Se verifica a través de esta información si el solicitante no posee mora de alguno de los impuestos que se cancelan al cementerio. Si es así deberá agregarse al cobro de la exhumación todos los impuestos en mora para poder brindar el servicio.
5. Se programa en el calendario de actividades el día y la hora, según la disponibilidad del personal para realizar la exhumación.
6. Se genera el formato ISAM para cobrar los servicios (según el catálogo de precios) que se han solicitado y luego cancelarse en caja.
7. Se imprime el formato ISAM para que el cliente pueda cancelar los servicios requeridos, proporcionando el recibo original al cliente y se guardan las demás copias para trabajo interno de la institución.

Flujo Alternativo:

1. El sistema hace verificación de los datos a guardar del registro, en caso que no sean correctos o que los campos requeridos estén vacíos envía un mensaje para que los corrija.

Poscondiciones:

1. Según el reglamento de la ley general de cementerios, las horas hábiles para inhumaciones serán de las siete a las dieciocho horas. A menos que la CSJ ordene que se haga en algún caso especial este servicio y se deberán seguir todas las medidas de seguridad que manda la ley y MSPAS en los casos de osamentas de personas con enfermedades infecto-contagiosas.

Nombre:	Generar títulos o actas de propiedad
Autor:	Luis Chávez
Fecha:	12/12/2010
Descripción: Permite imprimir títulos o actas de propiedad de las diferentes formas de inhumaciones que brinda el cementerio Municipal Santa Isabel, a fin de proporcionar a los usuarios del servicio un comprobante de la adquisición de los diferentes servicios proporcionados por la institución.	
Actores: 1. Atención al cliente 2. Cliente 3. Colector	
Precondiciones: Compra de una propiedad	
Flujo Normal: 1. Se solicitan los datos necesarios a través del formato para la compra de puestos a perpetuidad o la compra de fosas a e perpetuidad, según sea el caso. 2. Se genera el formato ISAM para cubrir los costos de lo que se desea comprar. 3. Posteriormente se generan los títulos o actas que comprueban la adquisición del servicio para los puestos a perpetuidad y para las fosas un acta que queda en archivo de la oficina y el recibo de formula ISAM que se le entrega al contribuyente para comprobar su compra, generándose con toda la información que ya se obtuvo del cliente en los casos anteriores.	
Flujo Alternativo: 1. El sistema hace verificación de los datos a guardar del registro, en caso que no sean correctos o que los campos requeridos estén vacíos envía un mensaje para que los corrija.	
Poscondiciones: 1. El título o acta es validada por el administrador del cementerio para ser firmada y sellada por las autoridades correspondientes otorgándole validez a la misma.	

4.2.2 DIAGRAMAS DE CLASES

CIInhumacion
Class

- Campos
 - DmCausasFallecimiento
 - DmCodEmpleado
 - DmCodInhumacion
 - DmCodIsam
 - DmCodPropiedad
 - DmDireccionFallecimiento
 - DmFechaInhumacion
 - DmLugarFallecimiento
 - DmNumeroPartidaDefuncion
 - DmSqlCon
- Propiedades
 - CausasFallecimiento
 - CodEmpleado
 - CodInhumacion
 - CodIsam
 - CodPropiedad
 - DireccionFallecimiento
 - FechaInhumacion
 - GetConexion
 - LugarFallecimiento
 - NumeroPartidaDefuncion
- Métodos
 - BuscarInhumacionCodigo
 - GuardarInhumacion
 - ListarInhumaciones
 - ModificarInhumacion
 - ObtenerDatosFallecido

CIClientes
Class

- Campos
 - DmApellidos
 - DmCodCliente
 - DmDireccion
 - DmEstadoCivil
 - DmFechaNacimiento
 - DmLugarExtensionDui
 - DmNombres
 - DmNumeroDui
 - DmSqlCon
 - DmTelefono
- Propiedades
 - Apellidos
 - CodCliente
 - Direccion
 - EstadoCivil
 - FechaNacimiento
 - GetConexion
 - LugarExtensionDui
 - Nombres
 - NumeroDui
 - Telefono
- Métodos
 - BuscarCliente
 - EliminarCliente
 - GuardarCliente
 - ListarClientes
 - ModificarCliente

CIPropiedad
Class

- Campos
 - DmAnchoPropiead
 - DmApellidosComprador
 - DmCalle
 - DmCategoria
 - DmCodCliente
 - DmCodEmpleado
 - DmCodIsam
 - DmCodPropiead
 - DmCodTitulo
 - DmCuadro
 - DmDireccionComprador
 - DmFechaCompra
 - DmFechaExtensionDuiComprador
 - DmFila
 - DmLargoPropiead
 - DmLugarExtesnsionDuiComprador
 - DmNombresComprador
 - DmNumeroCalleNorte
 - DmNumeroCalleOriente
 - DmNumeroCallePoniente
 - DmNumeroCalleSur
 - DmNumeroDuiComprador
 - DmNumeroNichosConstruir
 - DmNumeroPuestoNorte
 - DmNumeroPuestoOriente
 - DmNumeroPuestoPoniente
 - DmNumeroPuestoSur
 - DmOficioProfesionComprador
 - DmPuestoSepultura
 - DmSqlCon
 - DmTelefonoComprador
 - DmTipoPropiedad
 - DmZona
- Propiedades
 - AnchoPropiedad
 - ApellidosComprador
 - Calle
 - Categoria
 - CodCliente
 - CodEmpleado
 - CodIsam
 - CodPropiead
 - CodTitulo
 - Cuadro
 - DireccionComprador
 - FechaCompra
 - FechaExtensionDuiComprador
 - Fila
 - GetConexion
 - LargoPropiead
 - LugarExtesnsionDuiComprador
 - NombresComprador
 - NumeroCalleNorte
 - NumeroCalleOriente
 - NumeroCallePoniente
 - NumeroCalleSur
 - NumeroDuiComprador
 - NumeroNichosConstruir
 - NumeroPuestoNorte
 - NumeroPuestoOriente
 - NumeroPuestoPoniente
 - NumeroPuestoSur
 - OficioProfesionComprador
 - PuestoSepultura
 - TelefonoComprador
 - TipoPropiedad
 - Zona
- Métodos
 - BuscarPropiedadCodigo
 - GuardarPropiedad
 - ListarPropiead
 - ListarPropieadFallecido
 - VerificarPropiedad

CIEmpleado
Class

- Campos
 - DmApellidos
 - DmCodigoEmp
 - DmDepto
 - DmDireccion
 - DmDui
 - DmEstado
 - DmNit
 - DmNombres
 - DmSqlCon
 - DmTelefono
- Propiedades
 - ApellidoEmpleado
 - CodigoEmpl
 - DeptoEmpleado
 - DireccionEmpleado
 - DuiEmpleado
 - EstadoEmpleado
 - GetConexion
 - NitEmpleado
 - NombreEmpleado
 - TelefonoEmpleado
- Métodos
 - ActualizarEmpleado
 - GuardarEmpleado
 - Lista
 - MostrarEmpleados
 - ObtenerDptos

CIPartidaDefuncion
Class

- Campos
 - DmCausaMuerte
 - DmCodEmpleado
 - DmCodIsam
 - DmDireccionFallecido
 - DmEdad
 - DmEstadoCivil
 - DmGenero
 - DmHoraFechaFallecimiento
 - DmMedicoAvalaDefuncion
 - DmNombreFallecido
 - DmNombreMadre
 - DmNombrePadre
 - DmNombreSolicitante
 - DmNumeroPartidaDefuncion
 - DmOficioProfesionFallecidos
 - DmOrigenFallecido
 - DmSqlConec
 - DmTipoFallecimiento
- Propiedades
 - CausaMuerte
 - CodEmpleado
 - CodIsam
 - DireccionFallecido
 - Edad
 - EstadoCivil
 - Genero
 - GetConexion
 - HoraFechaFallecimiento
 - MedicoAvalaDefuncion
 - NombreFallecido
 - NombreMadre
 - NombrePadre
 - NombreSolicitante
 - NumeroPartidaDefuncion
 - OficioProfesionFallecidos
 - OrigenFallecido
 - TipoFallecimiento
- Métodos
 - BuscarPartidaDefuncion
 - DevolverDetalleIsam
 - GuardarPartidaDefuncion
 - ListarDefunciones
 - ModificarPartidaDefuncion
 - VerificarPartida

CIExhumacion
Class

- Campos
 - DmCausaInhumacion
 - DmCodEmpleado
 - DmCodExhumacion
 - DmCodInhumacion
 - DmFechaInhumacion
 - DmObservaciones
 - DmSqlConec
- Propiedades
 - CausaInhumacion
 - CodEmpleado
 - CodExhumacion
 - CodInhumacion
 - FechaInhumacion
 - GetConexion
 - Observaciones
- Métodos
 - BuscarInhumacionCodigo
 - GuardarExhumacion
 - ModificarExhumacion

CI Testigos
Class

- Campos
 - DmApellidos
 - DmCodTestigo
 - DmNombres
 - DmNumeroDui
 - DmNumeroPartidaDefuncion
 - DmParentescoFallecido
 - DmSqlCon
- Propiedades
 - Apellidos
 - CodTestigo
 - GetConexion
 - Nombres
 - NumeroDui
 - NumeroPartidaDefuncion
 - ParentescoFallecido
- Métodos
 - BuscarTestigo
 - EliminarTestigo
 - GuardarTestigo
 - ListarTestigos
 - ModificarTestigo

CI Conexion Cem...
Class

- Campos
 - DmConectar
 - DmDataBase
 - Dmestado
 - DmModulo
 - DmPassword
 - DmServidor
 - DmUsuario
- Propiedades
 - Estado
 - XDataBase
 - XPassword
 - XServidor
 - XUsuario
- Métodos
 - CerrarConexion
 - ConectoSQL

CI Funciones
Class

- Métodos
 - HabilitarBotones
 - HabilitarControles
 - InhabilitarBotones
 - InhabilitarControles
 - Letras
 - Limpiar
 - LimpiarGrid
 - NuevaHora
 - PoseeNumeros
 - TraducirEstadoCivil
 - VerificarControlesVacios

CIUsuarios
Class

- Campos
 - DmCodigoUsuario
 - DmEstado
 - DmPassword
 - DmRol
 - DmSqlCon
 - DmUsuario
- Propiedades
 - CodigoUsuario
 - Estado
 - GetConexion
 - Password
 - Rol
 - Usuario
- Métodos
 - GuardarUsuario
 - ModificarPasswordUsuario
 - ModificarUsuario
 - MostrarEmpleados
 - MostrarRoles
 - MostrarUsuario
 - StrToHash
 - UsuarioCorrecto
 - VerificarUsuario

CIPlan
Class

- Campos
 - DmAnio
 - DmCodigoPlan
 - DmDescripcion
 - DmNombre
 - DmSqlCon
- Propiedades
 - AnioPlan
 - CodigoPlan
 - DescripcionPlan
 - GetConexion
 - NombrePlan
- Métodos
 - GuardarPlan
 - ModificarPlan
 - MostrarPlanes
 - VerificarPlan

CIActividad
Class

Campos

- DmCodigoAct
- DmDescripcion
- DmEstado
- DmMotivo
- DmNombre
- DmSqlCon

Propiedades

- CodigoAct
- DescripcionAct
- Estado
- GetConexion
- Motivo
- NombreAct

Métodos

- ActividadEmpleado (+ 1 sobrecarga)
- EmpleadoActividad (+ 1 sobrecarga)
- GuardarActividad
- ModificarActividad
- MostrarActividades
- VerificarActividad

CIAP
Class

Campos

- DmCodigoActividad
- DmCodigoPlan
- DmEstado
- DmMotivo
- DmSqlCon

Propiedades

- CodigoActividad
- CodigoPlan
- EstadoActividad
- GetConexion
- Motivo

Métodos

- GuardarActividadAlPlan
- ModificarActividadDelPlan
- MostrarActividadesPorPlan
- MostrarActividadesPorPlanTodo

CIEP
Class

- Campos
 - DmCodigoEmpleado
 - DmCodigoPlan
 - DmEstado
 - DmMotivo
 - DmSqlCon
- Propiedades
 - CodigoEmpleado
 - CodigoPlan
 - Estado
 - GetConexion
 - Motivo
- Métodos
 - GuardarEmpleadoAlPlan
 - ModificarEmpleadoDelPlan
 - MostrarEmpleadosPorPlan
 - MostrarEmpleadosPorPlanTodo

CIEA
Class

- Campos
 - DmCodigoActividad
 - DmCodigoEA
 - DmCodigoEmpleado
 - DmEstado
 - DmFechaFin
 - DmFechaInicio
 - DmFinalizado
 - DmMotivo
 - DmSqlCon
- Propiedades
 - CodigoActividad
 - CodigoEA
 - CodigoEmpleado
 - Estado
 - FechaFin
 - FechaInicio
 - Finalizado
 - GetConexion
 - Motivo
- Métodos
 - CancelarActividad
 - FinalizarActividad
 - GuardarActividadEmpleado
 - ModificarFechaActividad
 - MostrarActividadesDeEmpleadNoProgreso
 - MostrarActividadesDeEmpleado
 - MostrarActividadesDelDia
 - MostrarActividadesEnProgreso
 - MostrarActividadesPendientes
 - MostrarActividadesRealizadas
 - VerificarActividadEmpleado

CIIsam
Class

- Campos
 - CodIsamVar
 - DmSqlCon
 - EstadoVar
 - FechaVar
 - IsdemVar
 - NombreClienteVar
 - NombreImpuesto...
 - NombreServicioVar
 - NumIsamVar
 - SerieVar
 - SubTotalVar
 - TotalVar
- Propiedades
 - GetConexion
 - ProCodigoIsam
 - ProEstado
 - ProFecha
 - ProIsdem
 - ProNombreCliente
 - ProNombreImp
 - ProNombreSer
 - ProNumIsam
 - ProSerie
 - ProTotal
 - ProTotalDetalleIs...
- Métodos
 - GuardarDetalleIs...
 - GuardarISAM
 - ListaImpuesto
 - ListarIsam
 - ListaServicio
 - ModificarTotal
 - MostrarDetallesIS...

CImpuesto
Class

- Campos
 - DmDescripcion
 - DmNombre
 - DmSqlCon
 - DmValor
- Propiedades
 - CodigoImpuesto
 - DescripcionImpu...
 - DmCodigoImpue...
 - GetConexion
 - NombreImpuesto
 - ValorImpuesto
- Métodos
 - ActualizarImpuesto
 - GuardarImpuesto
 - Lista

CServicios
Class

- Campos
 - DmDescripcion
 - DmNombre
 - DmSqlCon
 - DmValor
- Propiedades
 - CodigoServicio
 - DescripcionServicio
 - DmCodigoServicio
 - GetConexion
 - NombreServicio
 - ValorServicio
- Métodos
 - ActualizarServicio
 - GuardarServicio
 - Lista

CIEmpleado
Class

- Campos
 - DmApellidos
 - DmCodigoEmp
 - DmDepto
 - DmDireccion
 - DmDui
 - DmEstado
 - DmNit
 - DmNombres
 - DmSqlCon
 - DmTelefono
- Propiedades
 - ApellidoEmpleado
 - CodigoEmpl
 - DeptoEmpleado
 - DireccionEmpleado
 - DuiEmpleado
 - EstadoEmpleado
 - GetConexion
 - NitEmpleado
 - NombreEmpleado
 - TelefonoEmpleado
- Métodos
 - ActualizarEmpleado
 - EmpleadoActividad
 - GuardarEmpleado
 - Lista
 - MostrarEmpleados
 - ObtenerDptos

CIContratistas
Class

- Campos
 - DmApellidos
 - DmCodigo
 - DmDireccion
 - DmDui
 - DmEstado
 - DmNit
 - DmNombres
 - DmSqlCon
 - DmTelefono
- Propiedades
 - ApellidoContratista
 - CodigoContratista
 - DireccionContrati...
 - DuiContratista
 - EstadoContratista
 - GetConexion
 - NitContratista
 - NombreContratista
 - TelefonoContratista
- Métodos
 - ActualizarContrati...
 - GuardarContratista
 - Lista

CIHorario
Class

- Campos
 - DmCodEmpleado
 - DmDia
 - DmHoras
 - DmLlabe
 - DmSqlCon
- Propiedades
 - GetConexion
 - HorarioCodEmple...
 - HorarioDia
 - HorarioHoras
 - llabeHorario
- Métodos
 - ActualizarHorario
 - GuardarHorario
 - Lista
 - Lista2
 - MostrarEmpleados

CIDepartamento
Class

- Campos
 - DmCodigo
 - DmEncontrar
 - DmNombre
 - DmSqlCon
- Propiedades
 - CodigoDepto
 - GetConexion
 - NombreDeparta...
- Métodos
 - ActualizarDeparta...
 - GuardarDeparta...
 - Lista

CIVacaciones
Class

- Campos
 - DmCod
 - DmCodVaca
 - DmFechaFinal
 - DmFechaInicio
 - DmSqlCon
- Propiedades
 - CodigoEmpleado
 - CodigoVacacion
 - FechaFinal
 - FechaInicio
 - GetConexion
- Métodos
 - ActualizarVacacio ...
 - GuardarVacaciones
 - Lista
 - Lista2
 - MostrarEmpleados

4.3 DISEÑO DE INTERFACES DE ENTRADA Y SALIDA DE DATOS

AUTENTICACIÓN DE USUARIOS

Nombre de usuario

Contraseña

Aceptar Cancelar

SRC
VERSION 1.0

CAMBIO DE CONTRASEÑA

SANTA ISABEL
EL SALVADOR

UNIVERSIDAD DE EL SALVADOR
CATEDRA DE SISTEMAS

CAMBIO CONTRASEÑA

Datos de mi usuario

Mi usuario: CEMENTERIO

Contraseña actual:

Nueva contraseña:

Confirmar contraseña:

Opciones

Modificar Limpiar Salir

ACERCA DE...

Cementerio Santa Isabel

Versión 1.0

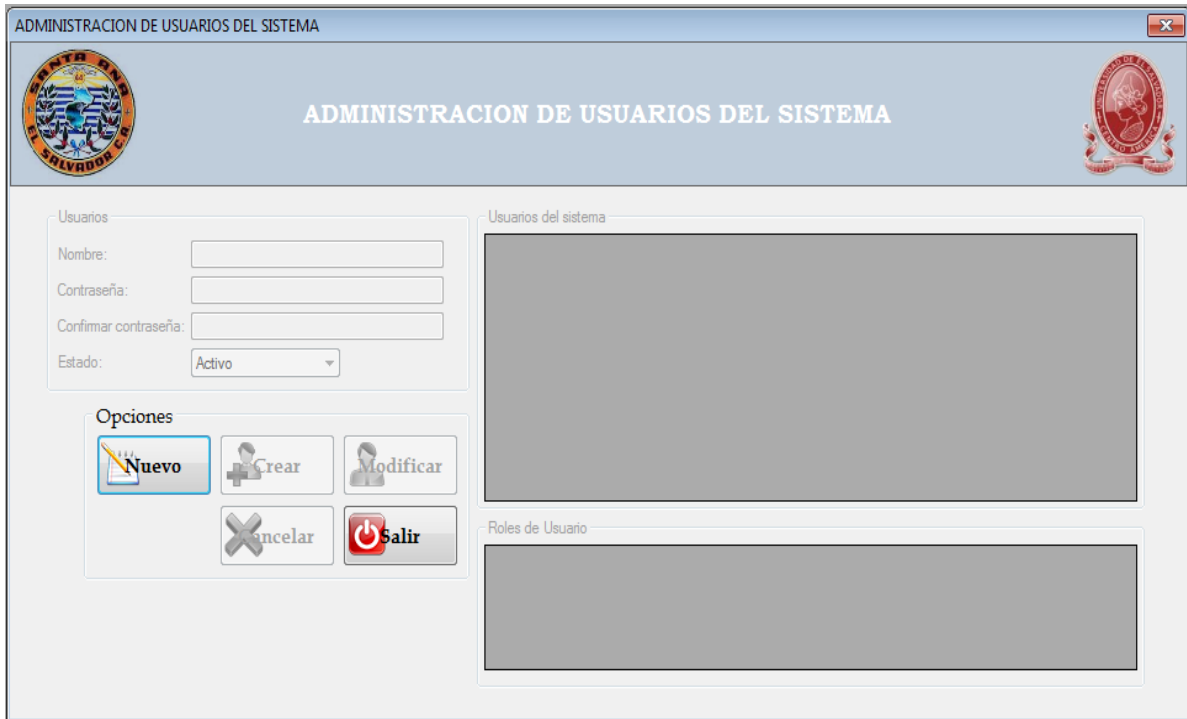
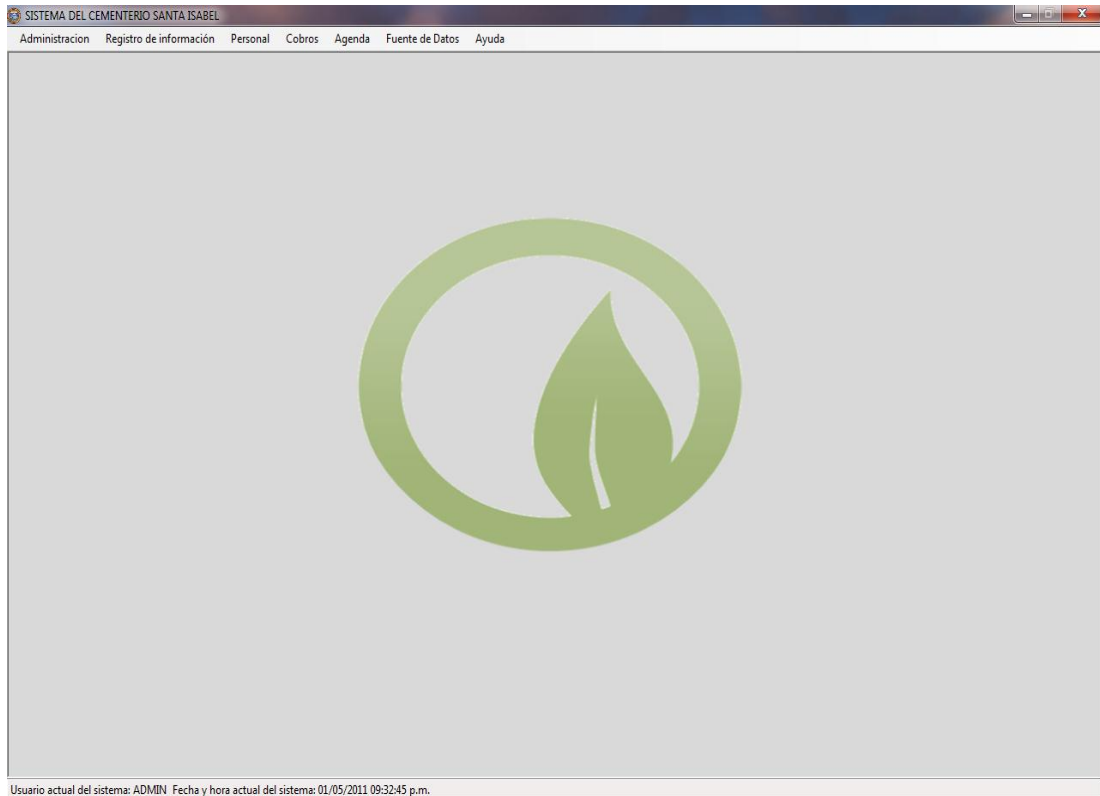
Derechos Reservados 2011

UES - FMO

Descripción:
Sistema encargado de la gestion de información del Cementerio Santa Isabel (CSI) de la ciudad de Santa Ana

Desarrolladores:

SRC
VERSION 1.0



ADMINISTRACION DE PLANES DE TRABAJO/ACTIVIDADES

PLANES DE TRABAJO/ACTIVIDADES

Planes de Trabajo | **Actividades**

Plan de Trabajo

Nombre: Año:

Descripción:

Planes de Trabajo

Opciones

 **Nuevo**  **Guardar**  **Modificar**  **Cancelar**  **Salir**

ADMINISTRACION DE PLANES DE TRABAJO/ACTIVIDADES

PLANES DE TRABAJO/ACTIVIDADES

Planes de Trabajo | **Actividades**

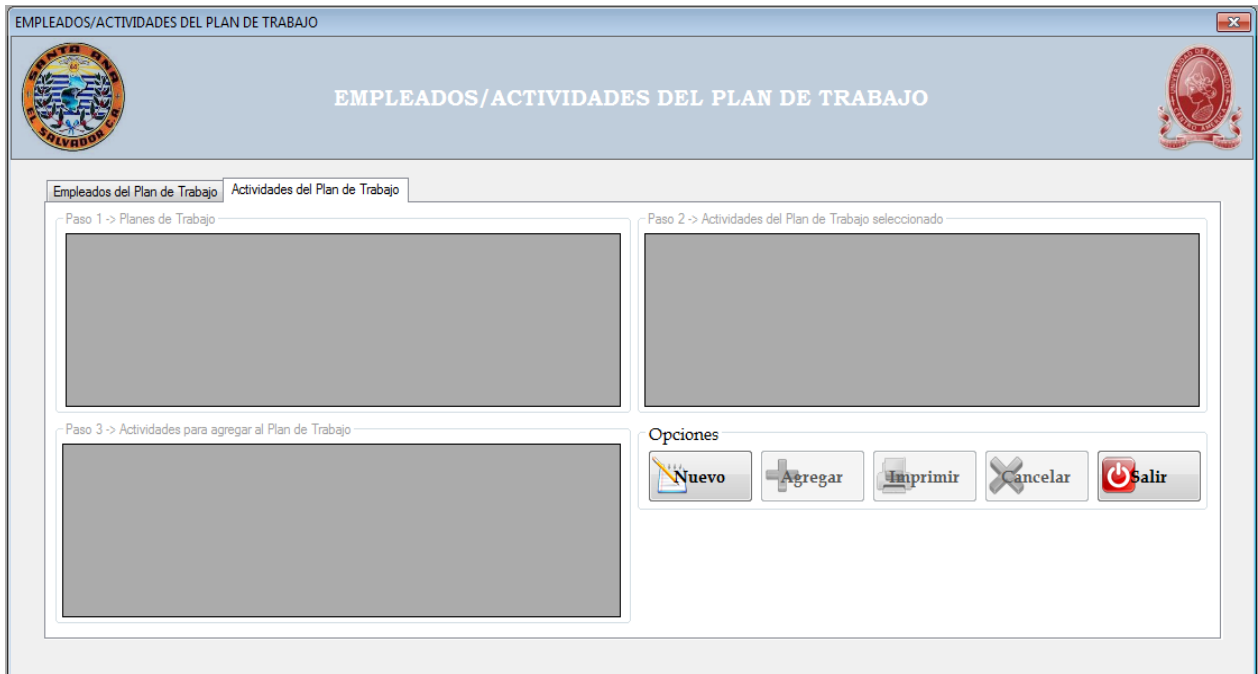
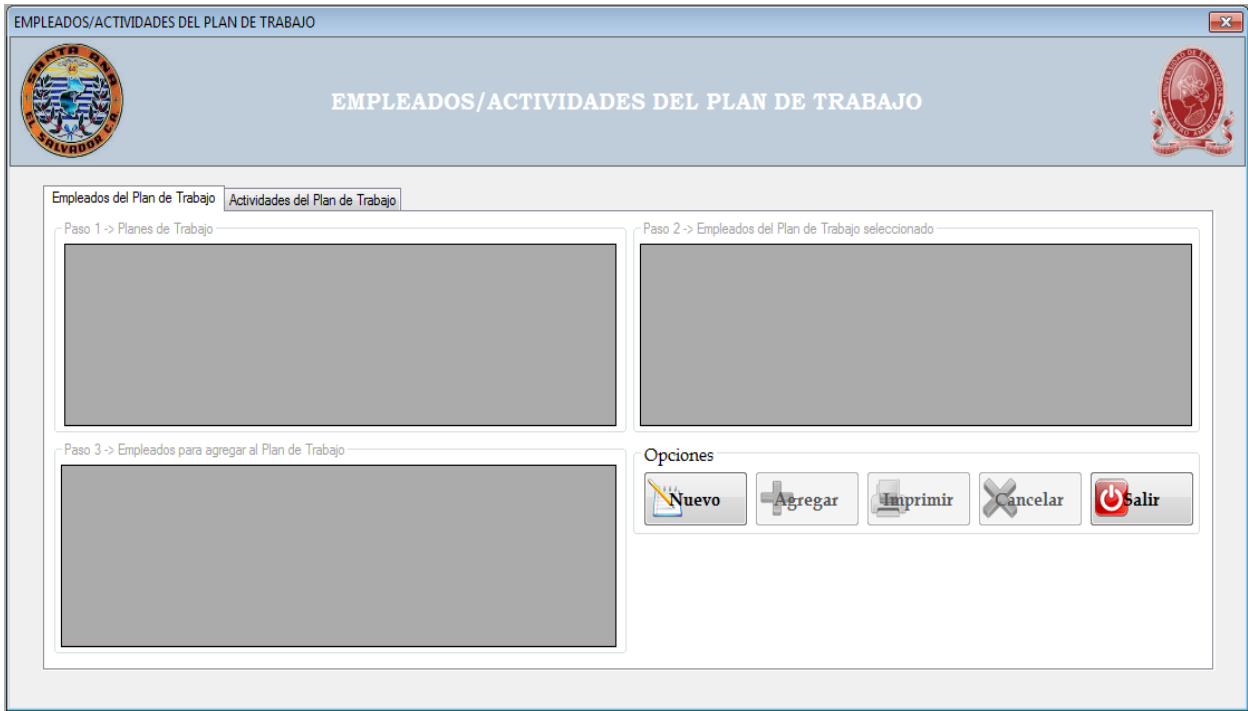
Actividad

Nombre: Descripción:


Listado de Actividades

Opciones

 **Nuevo**  **Guardar**  **Modificar**  **Imprimir**  **Cancelar**  **Salir**



ADMINISTRACION DE ACTIVIDADES DEL EMPLEADO



ADMINISTRACION DE ACTIVIDADES DEL EMPLEADO



Paso 1 -> Planes de Trabajo

Paso 2 -> Empleados del Plan de Trabajo seleccionado


Paso 3 -> Actividades del Plan de Trabajo seleccionado

30/03/2011


Paso 4 -> Actividades pendientes del empleado seleccionado

30/03/2011


Opciones

 Nuevo
 Signar
 Modificar
 Cancelar
 Salir

CONTROL DE ACTIVIDADES



CONTROL DE ACTIVIDADES






Actividades del día seleccionado
Actividades en progreso
Actividades pendientes entre dos fechas
Actividades realizadas entre dos fechas


Actividades que inician en la fecha establecida

Opciones


Fecha de actividad:
30/03/2011

 Mostrar
 Imprimir
 Salir

CONTROL DE ACTIVIDADES

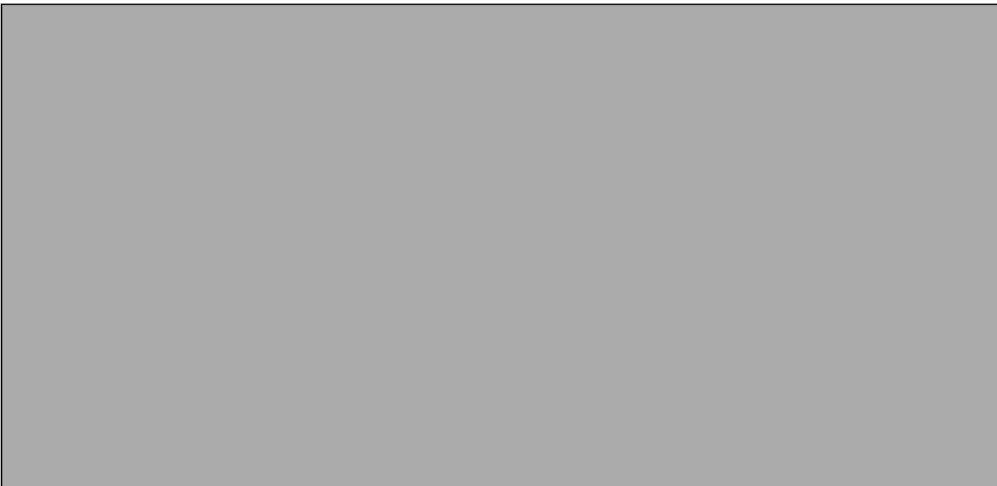


CONTROL DE ACTIVIDADES



Actividades del día seleccionado | Actividades en progreso | Actividades pendientes entre dos fechas | Actividades realizadas entre dos fechas


Actividades en progreso actualmente




Opciones

- Mostrar
- Imprimir
- Salir

CONTROL DE ACTIVIDADES




CONTROL DE ACTIVIDADES



Actividades del día seleccionado | Actividades en progreso | Actividades pendientes entre dos fechas | Actividades realizadas entre dos fechas

Actividades pendientes entre dos fechas establecidas




Opciones

Fecha de Inicio:
30/03/2011


Fecha de Fin:
30/03/2011

- Mostrar
- Imprimir
- Salir

CONTROL DE ACTIVIDADES



CONTROL DE ACTIVIDADES



Actividades del día seleccionado | Actividades en progreso | Actividades pendientes entre dos fechas | **Actividades realizadas entre dos fechas**

Actividades realizadas entre dos fechas establecidas

Opciones

Fecha de Inicio:
30/03/2011


Fecha de Fin:
30/03/2011

Mostrar


Imprimir

Salir

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS



ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS



Actividades | **Actividades del plan** | Empleados del plan | Actividades de empleados

Listado de Actividades

	Nombre de la actividad	Descripción de la actividad



Opciones

Motivo de la eliminación:

Eliminar

Salir

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

Actividades | Actividades del plan | Empleados del plan | Actividades de empleados

Planes de Trabajo

Nombre del Plan	Descripción del Plan	Año
[Placeholder for Plan List]		



Actividades del Plan de Trabajo seleccionado

[Placeholder for Selected Plan Activities]		
--	--	--

Opciones

Motivo de la eliminación:

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

Actividades | Actividades del plan | Empleados del plan | Actividades de empleados

Planes de Trabajo

Nombre del Plan	Descripción del Plan	Año
[Placeholder for Plan List]		



Listado de Empleados del Plan seleccionado

[Placeholder for Selected Plan Employees]		
---	--	--

Opciones

Motivo de la eliminación:

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

ELIMINACION Y CANCELACION DE ACTIVIDADES/EMPLEADOS

Actividades | Actividades del plan | Empleados del plan | **Actividades de empleados**

Listado de Empleados

	Nombres del Empleado	Apellidos del Empleado
[Empty Table]		



Actividades pendientes del empleado seleccionado

[Empty Table]		
---------------	--	--

Opciones

Motivo de la cancelación:

FINALIZACION DE ACTIVIDADES

FINALIZACION DE ACTIVIDADES

Actividades en progreso

Finalizar	Nombres del empleado	Apellidos del empleado	Nombre de la actividad	Inicio
[Empty Table]				

Opciones

ADMINISTRACION DE DEPARTAMENTOS

ADMON DE DEPARTAMENTOS

Departamentos

Nombre del departamento

Departamentos ingresados

ADMINISTRACION DE EMPLEADOS

ADMINISTRACION DE EMPLEADOS

Datos

Departamento Estado

Nombres Apellidos

Dirección

Teléfono DUI NIT

Opciones

Nombres de empleados

ADMINISTRACION DE HORARIOS




ADMINISTRACION DE HORARIOS

Seleccione un empleado

Cantidad de horas por día

Días:

No. Horas:

Días asignados por empleado

Opciones

 **Nuevo**  **Agregar**

 **Modificar**  **Cancelar**

 **Salir**

ADMINISTRACION DE VACACIONES




ADMINISTRACION DE VACACIONES

Seleccione un empleado

Fechas

Fecha de Inicio:

Fecha de finalización:

Detalle de vacaciones


Opciones

 **Nuevo**  **Agregar**


 **Modificar**  **Cancelar**

 **Salir**

ADMINISTRACION DE CONTRATISTAS



ADMINISTRACION DE CONTRATISTAS



Datos personales

Nombres Apellidos

Estado


Direccion


Teléfono


DUI


NIT


Opciones

 Nuevo


 Guardar

 Modificar

 Cancelar

 Salir

Contratistas



ADMINISTRACION DE SERVICIOS



ADMINISTRACION DE SERVICIOS



Datos del Servicio

Nombre del servicio

Descripcion

Valor \$

Opciones

 Nuevo

 Agregar

 Modificar

 Salir

Servicios Ingresados



ADMINISTRACION DE IMPUESTOS

ADMINISTRACION DE IMPUESTOS

Datos del Impuesto

Nombre del Impuesto

Descripcion

Valor \$

Opciones

Nuevo

Agregar

Modificar

Salir

Servicios Ingresados

FACTURA ISAM

FACTURA ISAM

Datos del cliente

ISDEM Factura No.

Serie

Fecha

Ciudad

Ciudad

Total en \$

Opciones

Nuevo

Agregar


Anular

Limpiar

Salir

Detalle ISAM

Buscar Clientes ✕




BUSQUEDA DE CLIENTES




Buscar cliente

Nombre a buscar

Opciones

 **Limpiar**

 **Salir**

Cientes Encontrados

Nombres del cliente	Número DUI	Teléfono	Dirección
[Empty table area]			

IMPUESTOS Y SERVICIOS ✕



IMPUESTOS Y SERVICIOS



Seleccione impuestos y servicios

No. ISAM ISDEM

Serie

Servicios ▼ Impuestos ▼

Opciones

 **Agregar**

 **Salir**

Impuestos y Servicios agregados

Administración de Clientes



ADMINISTRACIÓN DE CLIENTES




Datos del Cliente

Nombres <input type="text"/>	No. DUI <input type="text"/>
Apellidos <input type="text"/>	Lugar Extensión DUI <input type="text"/>
Estado Civil <input type="text"/>	Fecha Nacimiento <input type="text" value="03/04/2011"/>
Dirección <input type="text"/>	Teléfono <input type="text"/>


Opciones

 Nuevo
 Guardar
 Modificar
 Buscar
 Cancelar
 Salir

Registro de Propiedades



REGISTRO DE PROPIEDADES



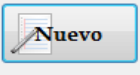



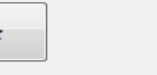
Datos Comprador

Codigo Isam <input type="text"/>	No. DUI <input type="text"/>	
Tipo de Propiedad <input type="text"/>	Lugar Extensión <input type="text"/>	
Código Título <input type="text"/> Código Cliente <input type="text"/>	Fecha Extensión <input type="text" value="03/04/2011"/>	
Nombres <input type="text"/>	Datos Propiedad	
Apellidos <input type="text"/>	Zona <input type="text"/>	Categoría <input type="text"/>
Oficio/Profesión <input type="text"/>	Cuadro <input type="text"/>	Calle <input type="text"/>
Teléfono <input type="text"/>	Num. Puesto/Sepultura <input type="text"/>	Ancho <input type="text"/> Metros
Dirección <input type="text"/>	Largo <input type="text"/> Metros	

Puesto

No. Nichos <input type="text" value="0"/>	Sepultura	
Puesto Nte. <input type="text"/>	Filea <input type="text"/>	Calle Nte. <input type="text"/>
Puesto Sur <input type="text"/>	Calle Sur <input type="text"/>	Calle Ote. <input type="text"/>
Puesto Pnte. <input type="text"/>	Calle Pte. <input type="text"/>	

Opciones


 Nuevo
 Guardar
 Modificar
 Cancelar
 Salir

Buscar ISAM

	CLIENTE	ISAM	ISDEM	SERIE	FECHA	TOTAL
▶						

Busqueda
 Código Cliente
 Parametro

Opciones


 Salir

Buscar Cliente

	Nombre Completo	Direccion	Dui	Lugar Extension	Telefono
▶					

Parámetro de Búsqueda
 Nombre

Opciones

 Salir

Crear Partida de Defunción

CREACIÓN DE PARTIDAS DE DEFUNCIÓN

Datos Solicitante

No. Partida defunción Número ISAM 

Nombre Solicitante

Datos Fallecido

Nombre Genero

Nombre Padre Dirección

Nombre Madre

Oficio/Profesión Fallecido

Estado Civil

Origen Fallecido

Edad

Causa Muerte

Tipo Fallecimiento

Hora Fallecimiento Horas Minutos AM PM

Fecha Fallecimiento

Medico Avalador

Opciones

 Nuevo  Guardar  Cancelar  Salir

Registro de Inhumación

REGISTRO DE INHUMACIONES

Datos Inhumación

Codigo ISAM  Partida Defunción 

Propiedad  Fecha Inhumación

Causas Fallecimiento

Lugar de Fallecimiento

Dirección Fallecimiento

Opciones

 Nuevo  Guardar  Cancelar  Salir

Busqueda de partida de defunción

Partida defuncion	Nombre del solicitante	Nombre del fallecido	Fecha fallecimiento	Nombre del pa
-------------------	------------------------	----------------------	---------------------	---------------

Parámetro de Búsqueda

Parametro Partida de defunción

Nombre

Opciones

Salir

Busqueda de inhumaciones

Partida defuncion	Nombre del fallecido	Nombre del padre	Nombre de la madre
-------------------	----------------------	------------------	--------------------

Parámetro de Búsqueda

Parametro Nombre del fallecido

Nombre

Opciones

Salir Salga del formula

Registrar Exhumación



REGISTRO DE EXHUMACIONES



Datos Exhumación

Codigo Exhumación Codigo Inhumación 

Fecha Exhumacion 

Causas

Observaciones

Opciones

 Nuevo  Guardar  Cancelar  Imprimir  Salir



PARTIDA No. _____

RECIBO No. _____

del domicilio de _____ originari _____ de
_____ nacionalidad _____
hij _____ de _____
falleció en _____ a las
_____ del día _____
_____ a consecuencias de _____
y sepultado en el Cementerio General de esta ciudad, el día de hoy, en _____
_____, No. _____, Calle _____, Cuadro _____
Médico asistente _____
SANTA ANA, _____ de _____
Dos mil _____

Custodio



PARTIDA No. _____

RECIBO No. _____

del domicilio de _____ originari _____ de
_____ nacionalidad _____
hij _____ de _____
falleció en _____ a las
_____ del día _____
_____ a consecuencias de _____
y sepultado en el Cementerio General de esta ciudad, el día de hoy, en _____
_____, No. _____, Calle _____, Cuadro _____
Médico asistente _____
SANTA ANA, _____ de _____
Dos mil _____

Custodio



PARTIDA DE DEFUNCIÓN

No. _____

A esta alcaldía se ha presentado el Sr.

Partida de defunción siguiente:

Nombre del Fallecido:

Edad

Estado Civil:

Originario de:

Domicilio:

Oficio:

Hijo de:

Enfermedad:

Falleció a las:

En el:

Médico Asistente:

Derechos Pagados:


Por Jefe del Cementerio

Por Jefe Registro Estado Familiar

Santa Ana, _____ de _____

ACTIVIDADES

1 de 1 100% Buscar | Siguiete



Cementerio Santa Isabel de la ciudad de Santa Ana


Reporte de todas las actividades del cementerio

Nombre de la actividad	Descripción de la actividad
ACTIVIDAD 1	PRIMERA ACTIVIDAD
ACTIVIDAD 2	SEGUNDA ACTIVIDAD
ACTIVIDAD 3	TERCERA ACTIVIDAD

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

ACTIVIDADES DEL PLAN DE TRABAJO

1 de 1 100% Buscar | Siguiete



Cementerio Santa Isabel de la ciudad de Santa Ana


Reporte de las actividades del plan de trabajo del año 2011

Nombre de la actividad	Descripción de la actividad
ACTIVIDAD 1	PRIMERA ACTIVIDAD
ACTIVIDAD 2	SEGUNDA ACTIVIDAD

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

ACTIVIDADES EN PROGRESO

1 de 1 100% Buscar | Siguiente



Cementerio Santa Isabel de la ciudad de Santa Ana


Reporte de las actividades en progreso

Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio
JUAN JOSE	SALAZAR	ACTIVIDAD 1	03/04/2011

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

ACTIVIDADES DE LA FECHA ESTABLECIDA

1 de 1 100% Buscar | Siguiente



Cementerio Santa Isabel de la ciudad de Santa Ana


Reporte de actividades que inician en la fecha 03/04/2011

Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio
JUAN JOSE	SALAZAR	ACTIVIDAD 1	03/04/2011

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

ACTIVIDADES PENDIENTES

1 de 1 100% Buscar | Siguiente



Cementerio Santa Isabel de la ciudad de Santa Ana


Reporte de las actividades pendientes entre las fechas 03/04/2011 y 06/04/2011

Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio
RAFAEL ANTONIO	CANO LINO	ACTIVIDAD 2	05/04/2011

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

ACTIVIDADES FINALIZADAS

1 de 1 100% Buscar | Siguiente



Cementerio Santa Isabel de la ciudad de Santa Ana

Reporte de las actividades finalizadas entre las fechas 30/03/2011 y 03/04/2011

Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio	Fecha de Fin
JUAN JOSE	SALAZAR	ACTIVIDAD 1	03/04/2011	03/04/2011

Fecha del reporte: 03/04/2011 Página 1 Reporte generado por: CEMENTERIO

4.4 DISEÑO DE LA BASE DE DATOS

4.4.1 CREACIÓN CONTENEDORES DE INFORMACIÓN (CREACIÓN DE TABLAS)

A continuación se presentan los diseños utilizados para las tablas que se utilizarán en el sistema, tal como se ha explicado en este capítulo se han dividido por módulos por lo que se presentan las tablas que utiliza cada módulo y posteriormente se presenta el diagrama de Entidad-Relación de la base de datos.

Una vez identificadas todas las tablas y columnas que necesita la base de datos, se determinara el tipo de dato de cada campo.

Existen cuatros categorías principales que pueden aplicarse prácticamente a cualquier aplicación de bases de datos:


- ✚ Texto
- ✚ Números
- ✚ Fecha y hora
- ✚ Contenido binario

Cada uno de éstos presenta sus propias variantes, por lo que la elección del tipo de dato correcto no sólo influye en el tipo de información que se puede almacenar en cada campo, sino que afecta al rendimiento global de la base de datos.

A continuación se muestran algunos de los lineamientos que se tomaron en cuenta para la elección de los datos adecuados para las diferentes tablas del proyecto.

- ✚ Identificar si una columna debe ser de tipo texto, numérico o de fecha o binario.
- ✚ Elegir el subtipo más apropiado para cada columna.
- ✚ Definir si puede contener o no un valor nulo.
- ✚ Configurar la longitud máxima para las columnas de texto y numéricas, así como otros atributos.
- ✚ Evaluación del uso de la generación automática de códigos.


TABLAS DEL MÓDULO DE USUARIOS (MODUSU)



MSTUSUARIOS (MODUSU)	
	CodUsuario
	Usuario
	Password
	Estado
	Rol




CATROLES (MODUSU)	
	CodRol
	Nombre
	Descripcion
	Estado

TABLAS DEL MÓDULO DE AGENDA (MODAGA)

TBLDETALLES (MODAGA)	
	CodPlan
	CodActividad
	Motivo
	Estado

MSTACTIVIDAD (MODAGA)	
	CodActividad
	Nombre
	Descripcion
	Motivo
	Estado

TBLPLEM (MODAGA)	
	CodPlan
	CodEmpleado
	Motivo
	Estado

TBLEMPLACT (MODAGA)	
	CodEA
	CodEmpleado
	CodActividad
	FechaInicio
	FechaFin
	Finalizado
	Motivo
	Estado

MSTPLAN (MODAGA)	
	CodPlan
	Nombre
	Descripcion
	Anio

TABLAS DEL MÓDULO DE PERSONAL (MODPER)

CATDEPARTAMENTO (MODPER)	
	CodDepto
	Nombre

MSTCONTRATISTA (MODPER)	
	CodContratista
	Nombres
	Apellidos
	Direccion
	Telefono
	Dui
	Nit
	Estado

MSTEMPLEADOS (MODPER)	
	CodEmpleado
	CodUsuario
	CodDepto
	Nombres
	Apellidos
	Direccion
	Telefono
	Dui
	Nit
	Estado

TBLHORARIO (MODPER)	
	CodDia
	CodEmpleado
	Dia
	Hora

TBLVACACIONES (MODPER)	
	CodVacaciones
	CodEmpleado
	FechaInicio
	FechaFin

TABLAS DEL MÓDULO DE COBROS Y COSTOS (MODCYC)

MSTISAM (MODCYC)	
	CodIsam
	NumeroIsam
	Isdem
	Serie
	CodCliente
	Fecha
	Estado
	Total
	Motivo
	FechaHoraAnulacion
	UsuarioCreacionIsam
	UsuarioAnulacionIsam

TBLDETALLEISAM (MODCYC)	
	CodDetalle
	CodIsam
	Cantidad
	Tipo
	Nombre
	ValorUnitario
	SubTotal

MSTIMPUESTO (MODCYC)	
	CodImpuesto
	Nombre
	Descripcion
	Valor

MSTCLIENTES (MODRIN)	
	CodCliente
	Nombres
	Apellidos
	EstadoCivil
	Direccion
	NumeroDui
	LugarExtensionDui
	DptoExtDui
	FechaNacimiento
	Telefono

MSTSERVICIO (MODCYC)	
	CodServicio
	Nombre
	Descripcion
	TipoServicio
	Valor

TABLAS DEL MÓDULO DE REGISTRO DE INFORMACIÓN (MODRIN)

MSTBENEFICIARIOS (MODRIN)		TBLPARTIDADEFUNCION (MODRIN)		MSTPROPIEDAD (MODRIN)	
<input type="checkbox"/>	CodBeneficiario	<input type="checkbox"/>	NumeroPartidaDefuncion	<input type="checkbox"/>	CodPropiead
<input type="checkbox"/>	CodPropiedad	<input type="checkbox"/>	CodIsam	<input type="checkbox"/>	TipoPropiedad
<input type="checkbox"/>	Nombres	<input type="checkbox"/>	NombreSolicitante	<input type="checkbox"/>	CodIsam
<input type="checkbox"/>	Apellidos	<input type="checkbox"/>	NombreFallecido	<input type="checkbox"/>	CodTitulo
<input type="checkbox"/>	NumeroDui	<input type="checkbox"/>	Edad	<input type="checkbox"/>	CodCliente
<input type="checkbox"/>	Estado	<input type="checkbox"/>	EstadoCivil	<input type="checkbox"/>	CodEmpleado
		<input type="checkbox"/>	OrigenFallecido	<input type="checkbox"/>	NombresComprador
		<input type="checkbox"/>	DireccionFallecido	<input type="checkbox"/>	ApellidosComprador
		<input type="checkbox"/>	OficioProfesionFallecidos	<input type="checkbox"/>	OficioProfesionComprador
		<input type="checkbox"/>	NombrePadre	<input type="checkbox"/>	TelefonoComprador
		<input type="checkbox"/>	NombreMadre	<input type="checkbox"/>	DireccionComprador
		<input type="checkbox"/>	MedicoAvalaDefuncion	<input type="checkbox"/>	NumeroDuiComprador
		<input type="checkbox"/>	HoraFechaFallecimiento	<input type="checkbox"/>	LugarExtenscionDuiComprador
		<input type="checkbox"/>	CodEmpleado	<input type="checkbox"/>	FechaExtensionDuiComprador
		<input type="checkbox"/>	TipoFallecimiento	<input type="checkbox"/>	Zona
		<input type="checkbox"/>	CausaMuerte	<input type="checkbox"/>	Categoria
		<input type="checkbox"/>	Genero	<input type="checkbox"/>	Cuadro
				<input type="checkbox"/>	Calle
				<input type="checkbox"/>	PuestoSepultura
				<input type="checkbox"/>	Fila
				<input type="checkbox"/>	AnchoPropiead
				<input type="checkbox"/>	LargoPropiedad
				<input type="checkbox"/>	NumeroPuestoNorte
				<input type="checkbox"/>	NumeroCalleNorte
				<input type="checkbox"/>	NumeroPuestoSur
				<input type="checkbox"/>	NumeroCalleSur
				<input type="checkbox"/>	NumeroPuestoOriente
				<input type="checkbox"/>	NumeroPuestoSur
				<input type="checkbox"/>	NumeroCalleSur
				<input type="checkbox"/>	NumeroPuestoOriente
				<input type="checkbox"/>	NumeroCalleOriente
				<input type="checkbox"/>	NumeroPuestoPoniente
				<input type="checkbox"/>	NumeroCallePoniente
				<input type="checkbox"/>	NumeroNichosConstruir
				<input type="checkbox"/>	FechaCompra

MSTISAM (MODCYC)		MSTCLIENTES (MODRIN)	
<input type="checkbox"/>	CodIsam	<input type="checkbox"/>	CodCliente
<input type="checkbox"/>	NumeroIsam	<input type="checkbox"/>	Nombres
<input type="checkbox"/>	Isdem	<input type="checkbox"/>	Apellidos
<input type="checkbox"/>	Serie	<input type="checkbox"/>	EstadoCivil
<input type="checkbox"/>	CodCliente	<input type="checkbox"/>	Direccion
<input type="checkbox"/>	Fecha	<input type="checkbox"/>	NumeroDui
<input type="checkbox"/>	Estado	<input type="checkbox"/>	LugarExtensionDui
<input type="checkbox"/>	Total	<input type="checkbox"/>	FechaNacimiento
		<input type="checkbox"/>	Telefono

TBLINHUMACION (MODRIN)	
<input type="checkbox"/>	CodInhumacion
<input type="checkbox"/>	CodIsam
<input type="checkbox"/>	NumeroPartidaDefuncion
<input type="checkbox"/>	CodPropiedad
<input type="checkbox"/>	CodEmpleado
<input type="checkbox"/>	CausasFallecimiento
<input type="checkbox"/>	LugarFallecimiento
<input type="checkbox"/>	DireccionFallecimiento
<input type="checkbox"/>	FechaInhumacion

MSTIMPUESTO (MODCYC)	
?	CodImpuesto
	Nombre
	Descripcion
	Valor

CATFIRMAS (MODRIN)	
?	CodFirmas
	FirmaAlcalde
	FirmaSecretario
	FirmaAdministrador
	FechaIngreso
	CodUsuario
	Activa

MSTSERVICIO (MODCYC)	
?	CodServicio
	Nombre
	Descripcion
	TipoServicio
	Valor

MSTTESTIGOS (MODRIN)	
?	CodTestigo
	NumeroPartidaDefuncion
	Nombres
	Apellidos
	NumeroDui
	ParentescoFallecido

MSTCLIENTES (MODRIN)	
?	CodCliente
	Nombres
	Apellidos
	EstadoCivil
	Direccion
	NumeroDui
	LugarExtensionDui
	DptoExtDui
	FechaNacimiento
	Telefono

MSTISAM (MODCYC)	
?	CodIsam
	NumeroIsam
	Isdem
	Serie
	CodCliente
	Fecha
	Estado
	Total
	Motivo
	FechaHoraAnulacion
	UsuarioCreacionIsam
	UsuarioAnulacionIsam

MSTPROPIEDAD (MODRIN)	
?	CodPropiedad
	TipoPropiedad
	CodIsam
	CodTitulo
	CodActa
	CodCliente
	CodEmpleado
	NombresComprador
	ApellidosComprador
	OficioProfesionComprador
	TelefonoComprador
	DireccionComprador
	NumeroDuiComprador
	LugarExtensionDuiComprador
	DptoExtDui
	FechaExtensionDuiComprador
	Zona
	Categoria
	Cuadro
	Calle
	NoCalle
	LetraCalle
	PuestoSepultura
	Fila
	AnchoPropiedad
	LargoPropiedad
	NumeroPuestoNorte
	NumeroCalleNorte
	NumeroPuestoSur
	NumeroCalleSur
	NumeroPuestoOriente
	NumeroCalleOriente
	NumeroPuestoPoniente
	NumeroCallePoniente
	NumeroNichosConstruir
	FechaCompra
	Nula
	FechaNulidad
	UsuarioAnulador
	UsuarioAutoriza
	JustificacionNulidad
	CodFirmas

MSTBENEFICIARIOS (MODRIN)	
?	CodBeneficiario
	CodPropiedad
	Nombres
	Apellidos
	NumeroDui
	Estado

TBLDETALLEISAM (MODCYC)	
?	CodDetalle
	CodIsam
	Cantidad
	Tipo
	Nombre
	ValorUnitario
	SubTotal

TBLPARTIDADEFUNCION (MODRIN)	
?	NumeroPartidaDefuncion
	CodIsam
	NombreSolicitante
	NombreFallecido
	Edad
	EstadoCivil
	OrigenFallecido
	DireccionFallecido
	OficioProfesionFallecidos
	NombrePadre
	NombreMadre
	MedicoAvalaDefuncion
	HoraFechaFallecimiento
	CodEmpleado
	TipoFallecimiento
	CausaMuerte
	Genero

TBLINHUMACION (MODRIN)	
?	CodInhumacion
	CodIsam
	NumeroPartidaDefuncion
	CodPropiedad
	CodEmpleado
	CausasFallecimiento
	LugarFallecimiento
	DireccionFallecimiento
	FechaInhumacion

TBLEXHUMACIONES (MODRIN)	
?	CodExhumacion
	CodInhumacion
	FechaInhumacion
	Observaciones
	CausaInhumacion
	CodEmpleado

4.4.2 NORMALIZACIÓN DE LA BASE DE DATOS

La normalización son una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica, consistencia y redundancia de los datos. Cada regla está basada en la que le antecede.

Grados de normalización: Existen básicamente tres niveles de normalización: Primera Forma Normal (1FN), Segunda Forma Normal (2FN) y Tercera Forma Normal (3FN). Existe hasta la quinta forma normal y dos métodos más para obtener la normalización de una base de datos, sin embargo este proyecto ha utilizado solamente las primeras tres formas normales, ya que con esto se logra la normalización de la misma en su diseño.

En la tabla siguiente se describe brevemente en que consiste cada una de las reglas, y posteriormente se explican con más detalle.

<u>Regla</u>	<u>Descripción</u>
Primera Forma Normal (1FN)	Incluye la eliminación de todos los grupos repetidos.
Segunda Forma Normal (2FN)	Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK).
Tercera Forma Normal (3FN)	Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Primera Forma Normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.

Aplicar a una base de datos la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples, evitando crear columnas que representen los mismos datos. La normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender.

Segunda Forma Normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos.





Una vez alcanzado el nivel de la Segunda Forma Normal, se controlan la mayoría de los problemas de lógica. Se puede insertar un registro sin un exceso de datos en la mayoría de las tablas.

Tercera Forma Normal

Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Comentamos anteriormente que una dependencia transitiva es aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

Se puede decir que estos son los principales objetivos cumplidos al normalizar los datos:

-  Controlar la redundancia de la información.
-  Evitar pérdidas de información.
-  Capacidad para representar toda la información.
-  Mantener la consistencia de los datos.

CAPÍTULO V.- PLAN DE IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN

5.1 OBJETIVOS DEL PLAN DE IMPLEMENTACIÓN

General:

Establecer un plan para el proceso de implementación del Sistema de Información del Cementerio Santa Isabel, a fin que la transición de los procesos manuales, a digitales sea asimilada por los empleados que utilizarán el sistema.

Específicos:

- Definir las actividades a realizar para el proceso de implementación del sistema.
- Establecer los recursos humanos, materiales y tecnológicos.
- Aportar la documentación técnica necesaria, para el proceso de implementación del proyecto.
- Seleccionar la estrategia adecuada de implementación del sistema de información.

5.2 LISTADO DE ACTIVIDADES DEL PLAN DE IMPLEMENTACIÓN

Plan de implementación

Para poder cumplir con los objetivos del plan de implementación se recurrirá a dividir dicho proceso mediante etapas, mediante las cuales se preparará a la institución para la implementación y capacitación del personal utilizando la siguiente distribución de etapas:

1. ETAPA 1

Preparación de la organización.

2. ETAPA 2

Instalación y configuración del hardware.

3. ETAPA 3

Instalación y configuración del Sistema.

4. ETAPA 4

Capacitación.

Cada uno de estas etapas se estructurará de la siguiente manera: Objetivos, propósito, actividades, y recursos.

En donde:

- ✓ **Objetivo:** Define el objetivo en cada una de las etapas que se requieren para la implementación.
- ✓ **Propósito:** El fin es aprovechar al máximo el recurso con que se cuenta y del que se necesitará, esto estará planteado en cada una de las fases así como el cumplimiento de los objetivos.
- ✓ **Actividades de trabajo:** Las actividades se deben realizar en conjunto para obtener la implantación del sistema con éxito. Definiendo su descripción para una mayor amplitud de lo que se pretende realizar.
- ✓ **Asignación de recursos:** Se expondrán los recursos necesarios para llevar a cabo la implementación del Sistema tomando en cuenta el recurso humano y material previsto en la implementación.

Etapa 1: Preparación de la organización.

Objetivo:

Preparar a las diferentes personas que laboran en la institución para el desarrollo de la implementación, así como definir los cambios que se deberán realizar en la ejecución de los procesos administrativos. Así mismo la selección de la estrategia de conversión del sistema antiguo (Procesos manuales) al nuevo sistema (Sistema de Información), permitiendo un correcto desarrollo de las actividades del plan.

Propósito:

- ✚ Controlar el avance de cada una de las etapas de plan
- ✚ Preparar diferentes medidas a fin de evitar retrasos en el tiempo programado para duración del proyecto.

Actividades de trabajo:

- ✚ Nombramiento del Jefe del proyecto.
- ✚ Nombrar el Jefe de la ejecución.
- ✚ Definir los encargados editar y digitar la información.

Asignación de recursos:

- **Nombramiento del Jefe del proyecto:** será el responsable de velar por el buen desarrollo de las actividades del plan y dentro de la organización se encargará de llevar a cabo la implantación del proyecto.

- **Nombramiento del jefe de ejecución:** Tendrá a cargo la capacitación del personal.
- **Nombramiento de encargados editar y digitar la información:** Serán los encargados de ingresar la información al sistema.

Etapa2: Instalación de Hardware

Objetivo:

Configuración e Instalación de todos los equipos que serán involucrados en el proceso de implementación del sistema.

Propósito:

- ✚ Verificar la instalación y configuración de los equipos informáticos.
- ✚ Verificar el área de instalación.
- ✚ Garantizar la protección del equipo instalado.

Actividades de trabajo

- ✚ Ubicación del equipo informático y de red de comunicaciones.
- ✚ Actualización y montaje del servidor.
 - Asignación de la configuración de red.
- ✚ Configuración de las estaciones de trabajo.
 - Asignación de la configuración de red.

Asignación de recursos:

Descripción de paquetes y hardware necesario: Las actividades antes descritas deben llevarse a cabo tomando en cuenta las consideraciones hechas en el **CAPÍTULO 3.1 DEFINICIÓN DE REQUERIMIENTOS.**

Etapa 3: Instalación y configuración del Sistema.

Objetivo:

Permitirá configurar el sistema operativo del equipo que funcionará como servidor, la base de datos en el servidor y la configuración de las estaciones de trabajo.

Propósito:

- ✚ Verificar la configuración en general de los diferentes equipos informáticos.
- ✚ Verificar el correcto funcionamiento del Sistema de Información.

Actividades de trabajo

- ✚ Instalación del Gestor de Base de Datos.
- ✚ Configuración del Gestor de Base de Datos.
 - Creación de la estructura de datos.
 - Creación de procedimientos almacenados.
 - Creación de disparadores.
- ✚ Creación de cuentas de usuarios.
- ✚ Instalación del Sistema de Información.
- ✚ Configuración de directivas de Windows de las cuentas de usuarios.
- ✚ Pruebas

Asignación de recursos:Descripción de instalación y configuración de paquetes necesarios para el funcionamiento

- ✚ Instalación y configuración del sistema operativo de los equipos: En las estaciones de trabajo la creación de las cuentas de usuario que utilizarán el sistema con el perfil adecuado de manipulación (Cuentas Limitadas) y en el equipo que funcionará como servidor se creará solamente la cuenta de administración para evitar que este sea manipulado poniendo en riesgo la integridad de los datos.
- ✚ Instalación y configuración de la base de datos: Configuración del servidor SQL Server 2008 Express R2. Posteriormente se debe restaurar la base de datos compuesta de la estructura, procedimientos almacenados y disparadores.
- ✚ Configuración de las estaciones de trabajo: Instalación y configuración del Sistema de Información a través del instalador y la fijación de la configuración regional adecuada de las estaciones de trabajo.
Esta implementación se hará a través de la estrategia de conversión directa, ya que no existía un sistema de información digital.
- ✚ Pruebas de funcionamiento.

Etapa 4: Capacitación.

Objetivo:

Se proporcionará la capacitación a los diferentes usuarios que este tenga para su correcta utilización y familiarización, así mismo la introducción de la información, procesamiento y como mostrar las salidas útiles en el momento que se necesite, además se instruirá a la persona que estará en el área de mantenimiento del sistema, y la configuración de cada uno de los usuarios dentro de la base de datos y del sistema como tal.

Propósito:

- ✚ Brindar manuales de usuario a los futuros usuarios del sistema.
- ✚ Definir y desarrollar las diferentes actividades dentro de la capacitación.
- ✚ Brindar las herramientas necesarias para el aprendizaje de los futuros usuarios.

Actividades de trabajo:

- ✚ Documentar a los participantes en la capacitación.

Recursos a utilizar

Los recursos humanos y materiales para la capacitación están detallados a continuación:

Recurso Humano	Recurso Material
- Personal Administrativo	- Manual del usuario
- Encargado del Proyecto	- Estaciones de trabajo

Tabla 5.1 Recursos utilizados en la capacitación

CAPITULO VI.

DOCUMENTACIÓN DEL

SOFTWARE

6.1 MANUAL DE USUARIO DEL SISTEMA DE INFORMACIÓN

El siguiente manual le dará una pequeña descripción que le permitirá aprender el funcionamiento del sistema para el manejo de información en el cementerio Santa Isabel.

AUTENTICACIÓN

SPLASH DEL SISTEMA

Al iniciar el sistema, lo primero que aparecerá es una pantalla llamada splash, la cual aparece mientras el sistema se encuentra cargando.



Figura 6.1.1 Splash del sistema

LOGIN DEL SISTEMA

Luego de pasados unos segundos, aparece esta pantalla, la cual es la clave para el inicio de sesión, en la cual toda persona que desee ingresar debe registrar sus datos antes de poder ingresar al sistema, para ello el usuario que desee acceder debe de ingresar un nombre de usuario y contraseña que son propios de cada usuario.



Figura 6.1.2 Pantalla de autenticación

- A.** Acá se digita el nombre del usuario que corresponde a un empleado del sistema.
 - B.** En este campo debe de digitar la contraseña del usuario que está ingresando en el campo anterior, de lo contrario no podrá acceder al sistema.
 - C.** Botón aceptar (también se puede hacer clic en dicho botón con la tecla Enter) que permite validar los datos ingresados, y en caso de haber ingresado usuario y contraseña correctos, permite ingresar al sistema.
 - D.** Permite cancelar el proceso de autenticación, con lo cual se sale del sistema.
- Cuando se ingresa un nombre de usuario o contraseña incorrecta, o de un usuario que está inactivo, y hacemos clic en el botón aceptar, el sistema no deja que dicho usuario ingrese al sistema, y nos muestra el siguiente mensaje de error:

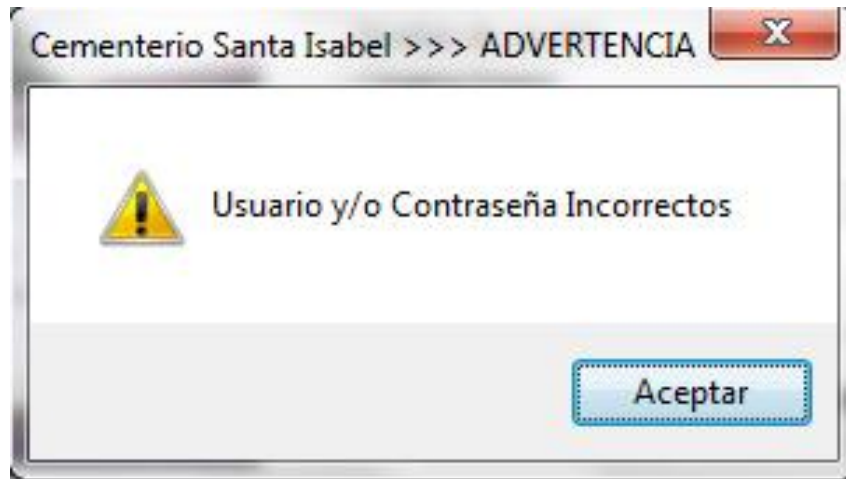


Figura 6.1.3 Error de autenticación

Ahora bien, si el usuario y la contraseña ingresados correctamente y corresponde a un usuario activo del sistema, cuando hacemos clic en el botón aceptar, el sistema nos permite ingresar al sistema y nos muestra el siguiente mensaje de bienvenida:

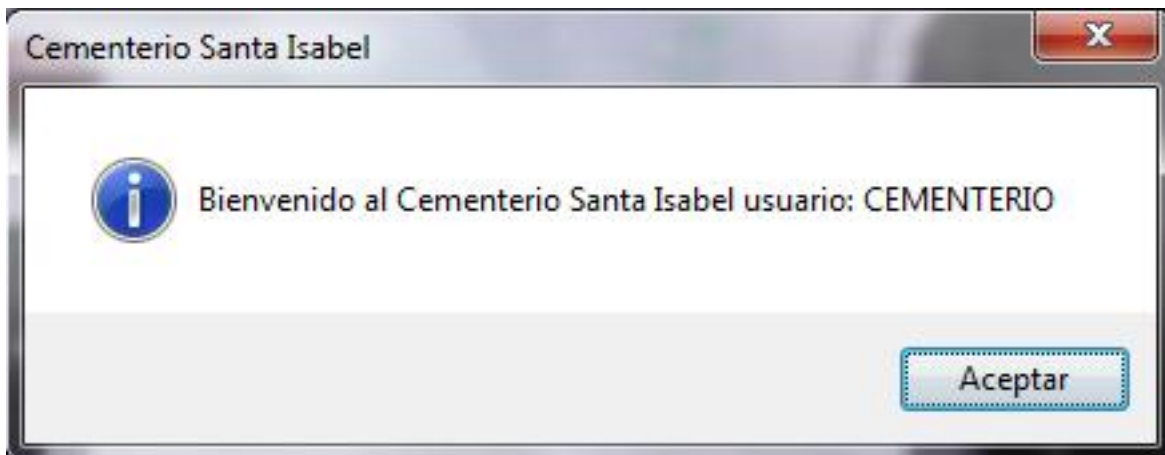


Figura 6.1.4 Bienvenida del sistema

MENU PRINCIPAL DEL SISTEMA

Luego de la bienvenida, ingresamos al menú principal, el cual contiene los diferentes menús para las diferentes opciones que tiene el sistema, pero el acceso a las mismas dependerá del rol de usuario con el que disponga el usuario ingresado. El menú principal tiene la siguiente presentación:

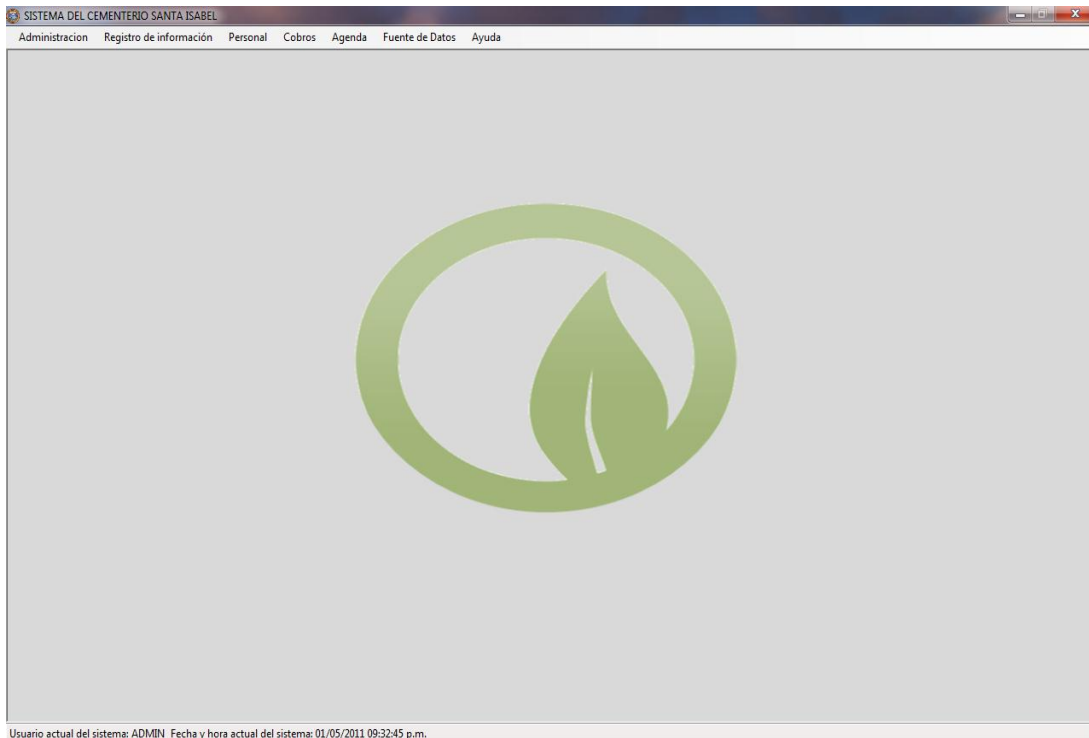


Figura 6.1.5 Menú principal

Este menú principal presenta las siguientes características que se describen a continuación:

1. Barra de título donde se presenta el nombre del sistema.

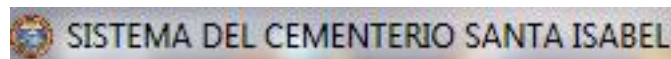


Figura 6.1.6 Barra de título

2. Barra de menús



Figura 6.1.7 Barra de menús

3. Barra de estado la cual presenta dos elementos:
 - a. El usuario en uso del sistema

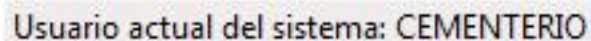


Figura 6.1.8 Usuario del sistema

- b. Fecha y hora actual del sistema

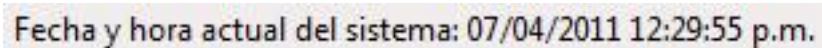


Figura 6.1.9 Fecha y hora del sistema

MENUS DEL MENÚ PRINCIPAL

Las opciones que poseen los menús son habilitadas de acuerdo al rol que posee el usuario que ha ingresado al sistema, es decir que mientras el rol tenga un nivel más bajo, el usuario tendrá acceso a menos opciones del sistema, ahora bien, vamos a explicar uno a uno las diferentes opciones con las que cuenta el sistema.

MENÚ ADMINISTRACIÓN

Permite administrar todo lo relacionado a la administración de usuarios, así como también al manejo de las sesiones del sistema, además del manejo de la cadena de conexión necesario por cuestiones de portabilidad.

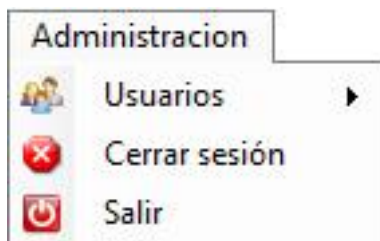


Figura 6.1.10 Administración

OPCIÓN USUARIOS

Permite administrar a los usuarios del sistema, brindando varias opciones sobre los usuarios, las cuales son:



Figura 6.1.11 Sub-opciones de usuarios

Sub-opción Crear y Modificar usuarios

Permite la creación de nuevos usuarios, así como también la modificación de usuarios existentes, incluyendo el estado de dichos usuarios; estos usuarios se relacionan directamente con los empleados del cementerio.

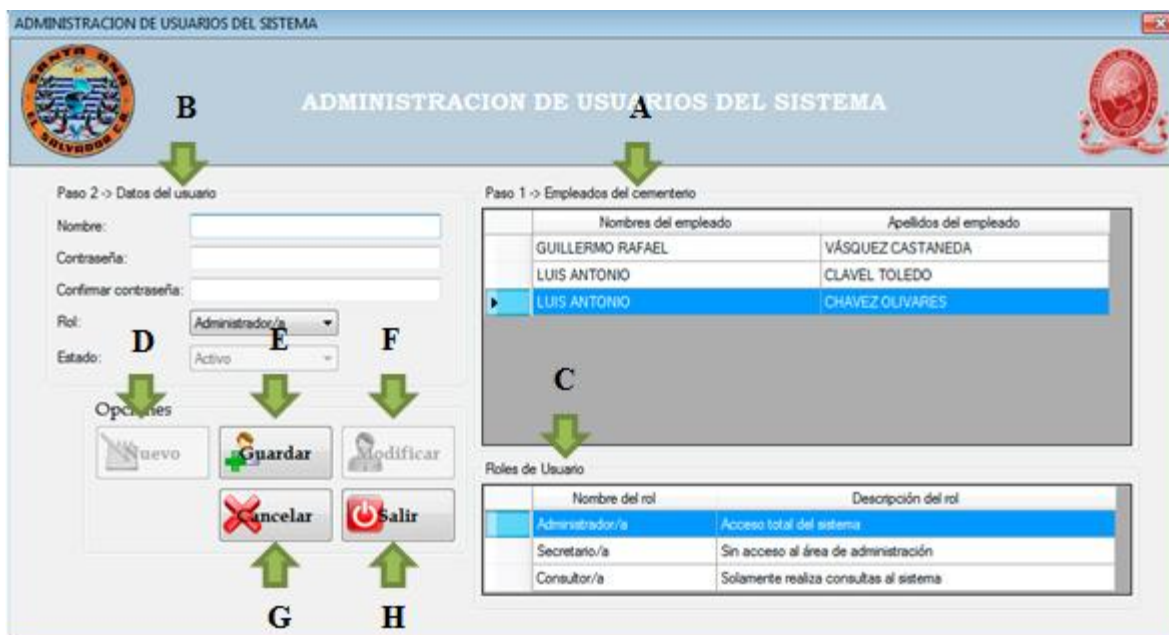


Figura 6.1.12 Administración de usuarios del sistema

A. Empleados del cementerio muestra un listado con todos los empleados activos que están registrados en el sistema, para poder crear un nuevo usuario al empleado, debe seleccionar al empleado, luego si el empleado no dispone de un usuario, el sistema automáticamente nos permitirá agregar los datos necesarios que se especifican en el siguiente ítem, de lo contrario el sistema nos permitirá poder modificar el usuario correspondiente al empleado seleccionado.

B. Los datos del usuario son los datos necesarios para poder registrar a un usuario dentro del sistema de información. Para poder registrarlo correctamente deberá llenar todos los campos solicitados correctamente, siguiendo las siguientes directrices:

- 1) El nombre y la contraseña así como la confirmación no pueden contener caracteres especiales.
- 2) El rol del usuario debe ser seleccionado del combobox, para poder tener una mejor idea, puede verificar que hace cada rol en listado inferior derecho de la

ventana. Cada rol tiene diferentes opciones de acceso las cuales se muestran a continuación:

- a. Administrador/a: Con este rol el usuario tiene acceso a todas las opciones del sistema, es el usuario con más privilegios dentro del sistema.
- b. Secretario/a: Este rol permite acceder a la mayoría de opciones dentro del sistema, pero no podrá acceder a aquellas opciones que son exclusivamente responsabilidad del administrador, por ejemplo finalizar las actividades, etc.
- c. Consultor/a: Este rol posee menos privilegios que los demás roles, ya que solamente le es permitido acceder a aquellas opciones que son puramente de consulta, no tiene la opción de agregar, modificar o eliminar nada dentro del sistema.

3) El estado del usuario a registrar será siempre por defecto activo, pudiendo ser inactivo en caso que el usuario sea modificado.

Si se diera el caso que el nombre de usuario que estamos ingresando ya existe, el sistema nos mostrará el siguiente mensaje de error:

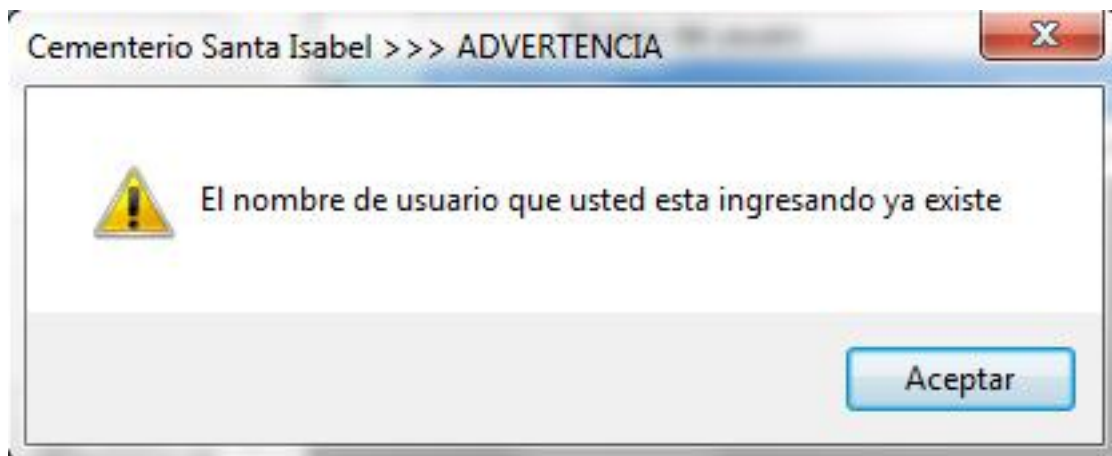


Figura 6.1.13 Mensaje de Error de nombre de usuario

En caso que se omita alguna de las directrices anteriores, obtendríamos el siguiente mensaje de error al presionar el botón guardar:

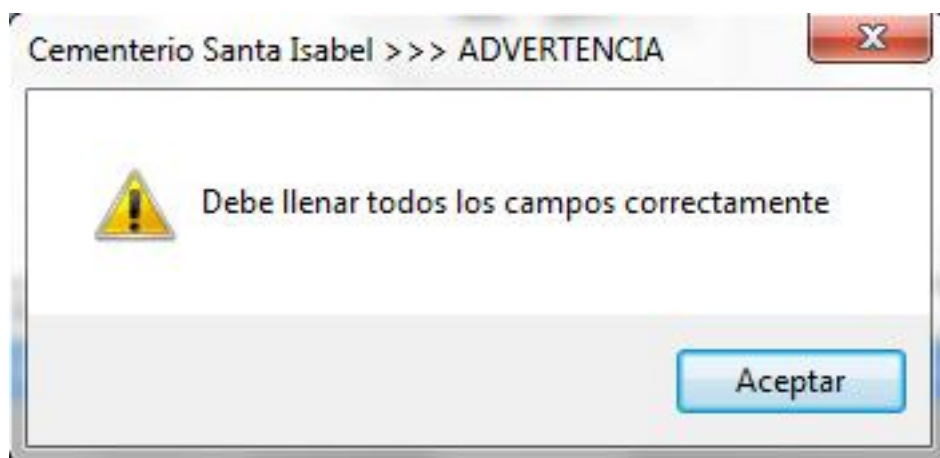


Figura 6.1.14 Mensaje de Error de Nuevo Usuario

C. Roles del usuario muestra un listado con todos roles que están disponibles para el usuario, este listado es solamente de información, no tiene ninguna otra funcionalidad.

D: Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.12 con el botón habilitado y los demás campos se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran el listado de los empleados y dependiendo del empleado seleccionado se habilitara o deshabilitara los campos necesarios para el ingreso de información del usuario.

E: Representa al botón guardar (se puede hacer clic en este botón con la tecla Enter), que deberá presionarse cuando hayamos proporcionado todos los datos necesarios para guardar un nuevo usuario o para modificar uno existente. En caso que hayamos seguido las directrices antes mencionadas, en el caso de un nuevo usuario, se mostrará en pantalla un mensaje preguntándonos si queremos guarda el usuario tal como se muestra en la siguiente imagen:

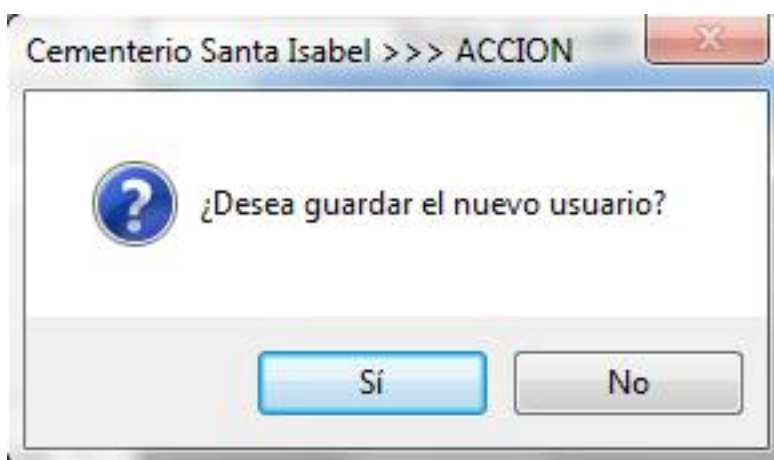


Figura 6.1.15 Mensaje de preguntando si queremos guardar el usuario

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que el usuario se guardó correctamente con el siguiente mensaje:

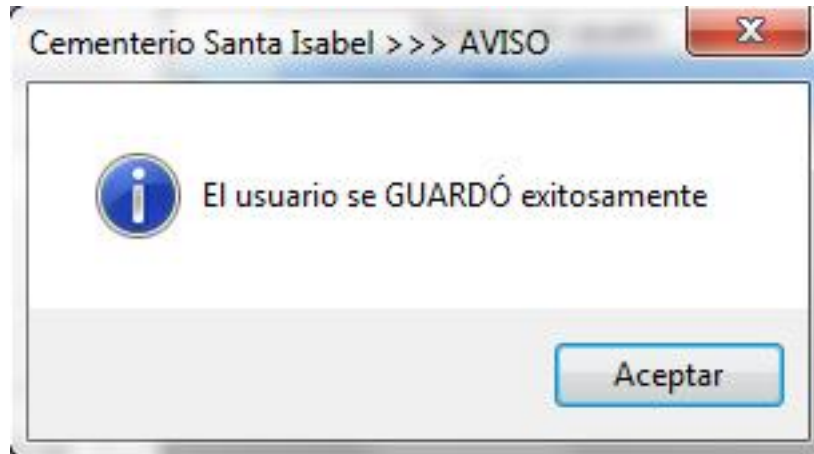


Figura 6.1.16 Mensaje confirmando al usuario guardado

Si el caso es la modificación de los datos de un usuario, se mostrara la siguiente pregunta en forma de mensaje:

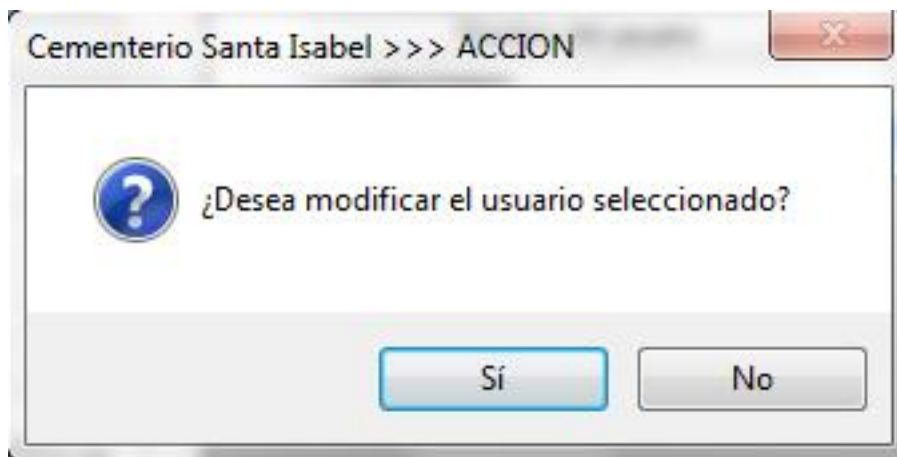


Figura 6.1.17 Mensaje preguntando si queremos modificar el usuario

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que el usuario se modificó correctamente con el siguiente mensaje:

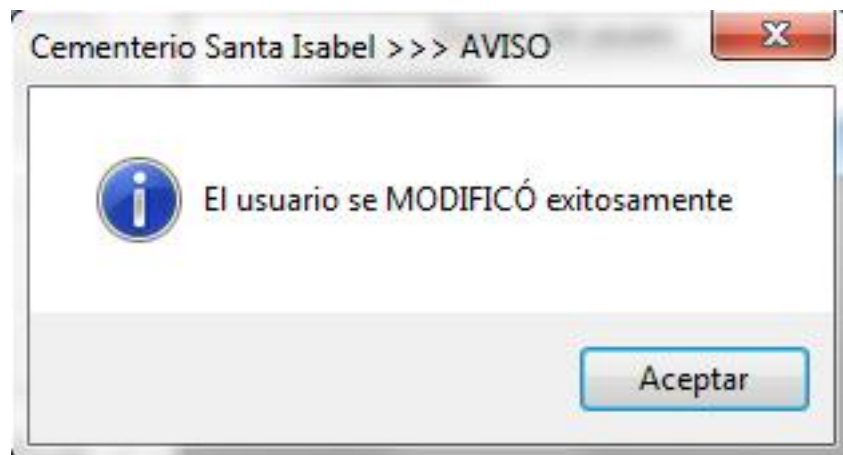


Figura 6.1.18 Mensaje confirmando al usuario modificado

F: Botón que permite guardar las modificaciones de un registro y que por defecto se encuentra inhabilitado, ya que para que este se encuentre activo primero deberá seleccionar algún empleado del listado y si este empleado ya posee un usuario, se habilitara este botón, luego si usted hace clic en este botón se habilitara la modificación de los datos del usuario (rol y estado), a la vez que se habilitara el botón guardar para poder guardar los cambios.

G: Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el siguiente mensaje:

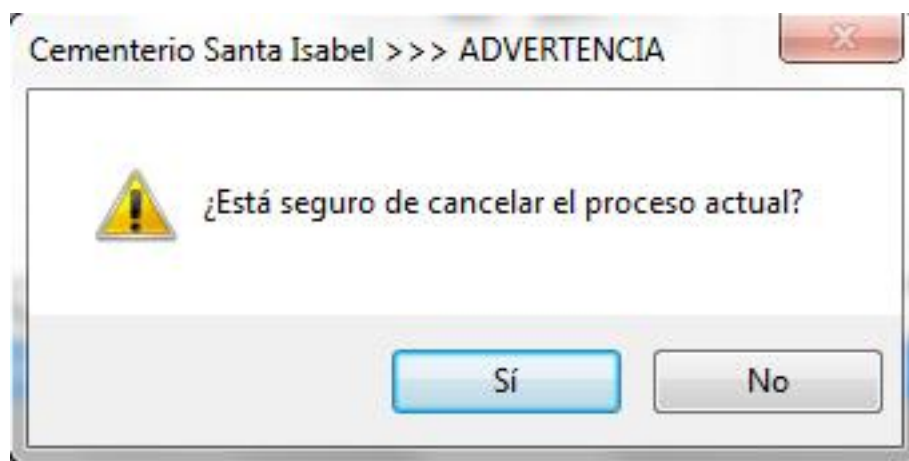


Figura 6.1.19 Mensaje preguntando si queremos cancelar la acción

H: Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la Figura 6.1.12, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán. En caso que queramos salir nos mostrara el siguiente mensaje:

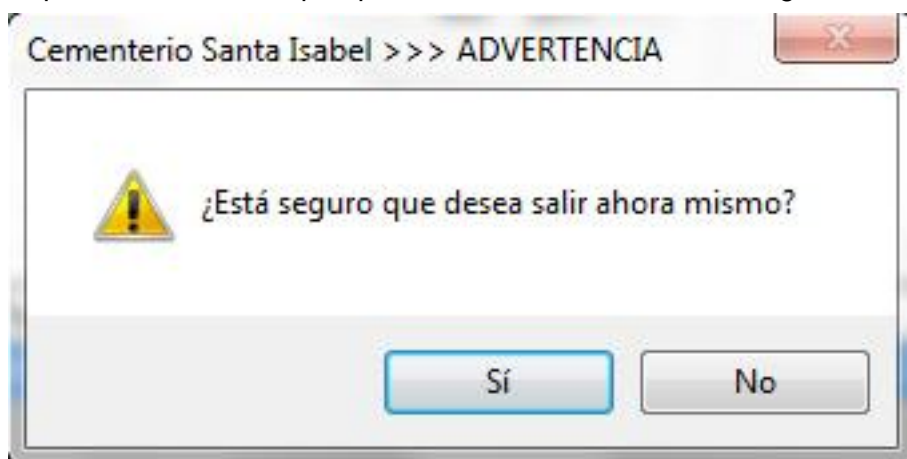


Figura 6.1.20 Mensaje preguntando si queremos salir del formulario

Sub-opción Modificar mi contraseña

Permite modificar la contraseña del usuario activo del sistema, es decir el usuario que ha iniciado sesión en el sistema.

The screenshot shows a web application window titled "CAMBIO DE CONTRASEÑA". At the top, there are two logos: the coat of arms of Santa Ana, El Salvador, and the coat of arms of the University of El Salvador. The main heading is "CAMBIO CONTRASEÑA". Below this, there is a section labeled "Datos del usuario activo" with a green arrow "A" pointing to it. This section contains four input fields: "Mi usuario:" with the value "ADMIN", "Contraseña actual:", "Nueva contraseña:", and "Confirmar contraseña:". Below the input fields is a section labeled "Opciones" with a green arrow "B" pointing to it. This section contains three buttons: "Guardar" (with a floppy disk icon), "Limpiar" (with a left-pointing arrow icon), and "Salir" (with a power button icon). Green arrows "C" and "D" point to the "Limpiar" and "Salir" buttons respectively.

Figura 6.1.21 Cambio de contraseña

A. Los datos del usuario activo son los datos necesarios para poder modificar o cambiar la contraseña del usuario que se encuentra activo en el sistema. Para poder registrarlo correctamente deberá llenar todos los campos solicitados correctamente, siguiendo las siguientes directrices:

- 1) Todas las contraseñas (actual, nueva, confirmación) no pueden contener caracteres especiales.

Si se da el caso que la contraseña actual que estamos ingresando es incorrecta el sistema nos mostrara el siguiente mensaje de error:

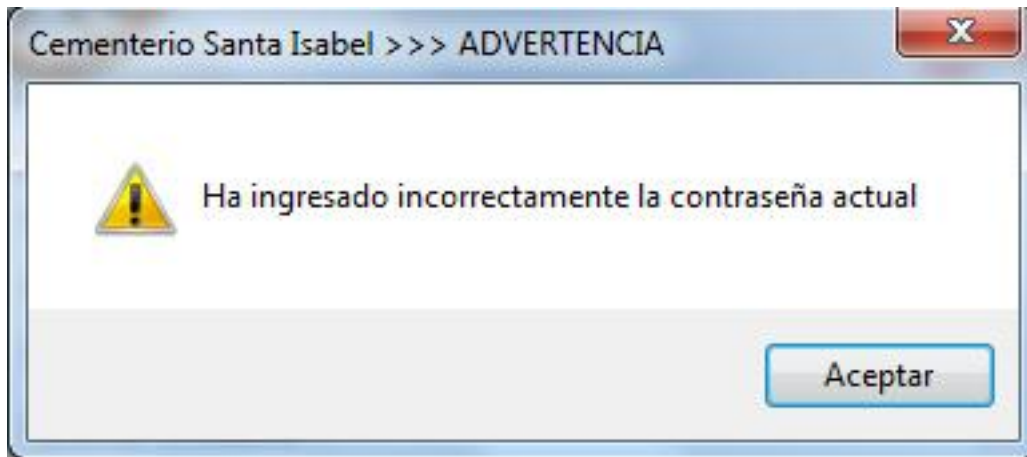


Figura 6.1.22 Mensaje de Error de contraseña actual

Ahora bien, si la nueva contraseña con su respectiva confirmacion no son iguales, el sistema nos mostrara el siguiente mensaje de error:

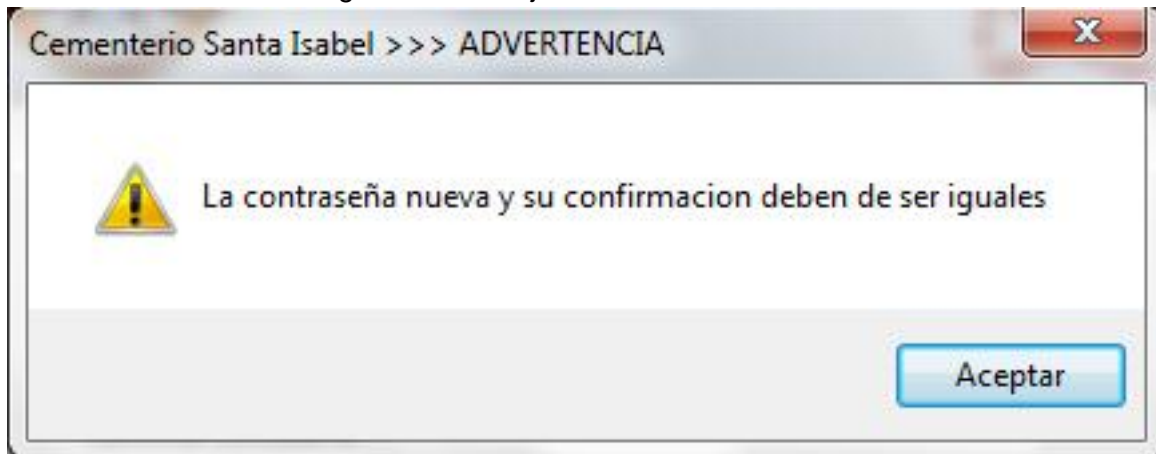


Figura 6.1.23 Mensaje de Error de confirmacion de contraseña

B: Representa al botón guardar (se puede hacer clic en este botón con la tecla Enter), que deberá presionarse cuando hayamos proporcionado todos los datos necesarios para para modificar la contraseña del usuario. En caso que hayamos seguido las directrices antes mencionadas, se mostrara un mensaje confirmándonos del cambio de contraseña:

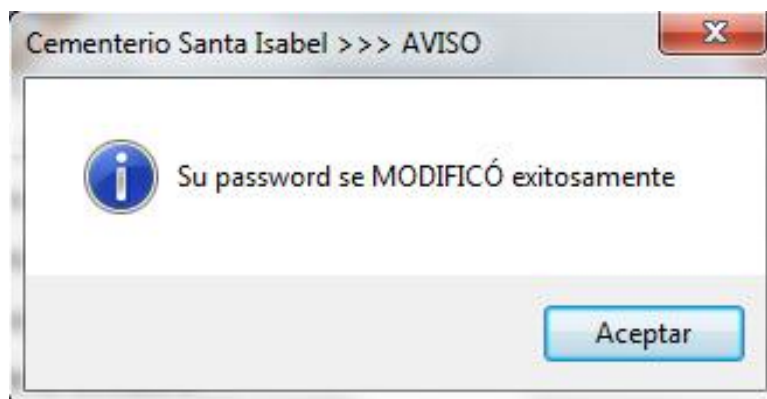


Figura 6.1.24 Mensaje confirmándonos cambio de contraseña

C: Este botón permite limpiar todos los campos del formulario, en otras palabras borra toda la información que hayamos ingresado.

D: Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la Figura 6.1.21, sin embargo debemos aclarar que si no hemos modificado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.20

OPCIÓN CERRAR SESIÓN

Permite cerrar la sesión del usuario activo, brindando la posibilidad de iniciar sesión con otro usuario, ya que cuando se cierra sesión, el sistema vuelve a la pantalla de login para poder iniciar sesión con otro usuario.

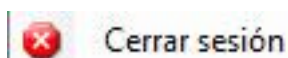


Figura 6.1.25 Cerrar sesión

Cuando se cierra sesión, el sistema nos pregunta si estamos seguros de cerrar la sesión actual del sistema (en caso de responder Sí cierra la sesión) tal como lo muestra la siguiente imagen:

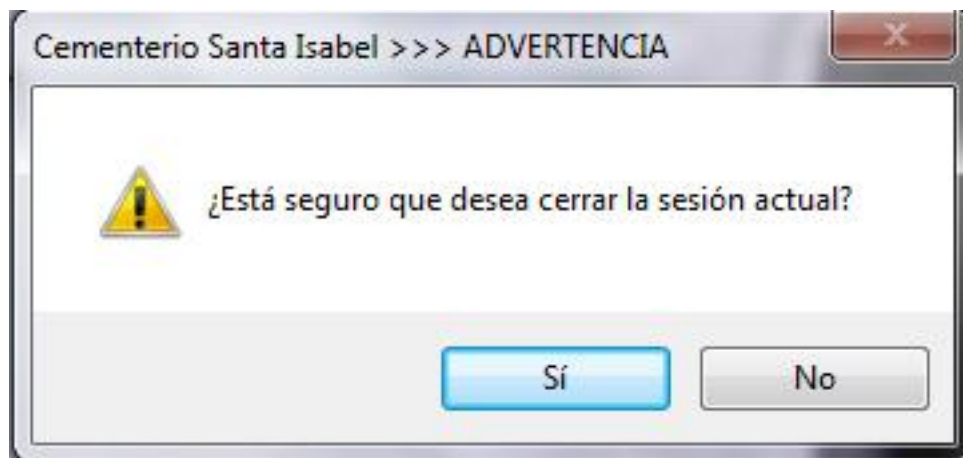


Figura 6.1.26 Advertencia sobre cerrar sesión

OPCIÓN SALIR

Permite salirse del sistema al igual que el botón superior derecho con una "X" del menú principal, es decir nos saca del sistema. Podemos acceder directamente a esta opción presionando la tecla ESC.

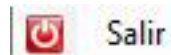


Figura 6.1.27 Salir del sistema

Cuando deseamos salir del sistema, este nos pregunta si estamos seguros de salir del mismo (en caso de responder SÍ se cierra todo el sistema) tal como lo muestra la siguiente imagen:

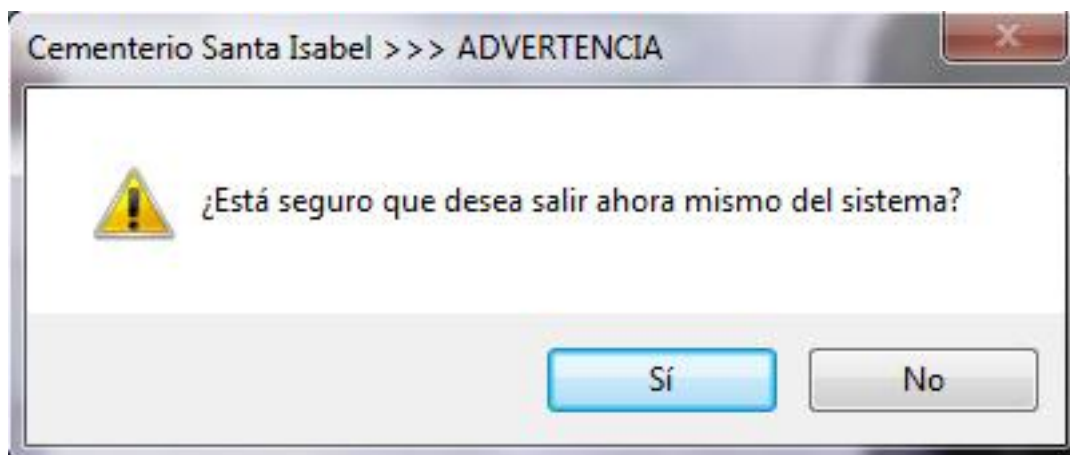


Figura 6.1.28 Advertencia sobre salir del sistema

MENÚ REGISTRO DE INFORMACIÓN

Permite administrar todo lo relacionado al registro de información, es decir clientes, propiedades dentro del cementerio, inhumaciones, partidas de defunción, etc.



Figura 6.1.29 Registro de información

OPCIÓN CLIENTES

Esta opción nos permite guardar y modificar clientes que se acerquen a las instalaciones del cementerio a fin de realizar una compra o a actualizar sus datos personales debido a que poseen una propiedad.



Figura 6.1.30 Administración de Clientes

A: Los datos del cliente son los datos necesarios para poder registrar a un cliente dentro del sistema de información. Para poder registrarlo correctamente deberá llenar todos los campos solicitados correctamente, siguiendo las siguientes directrices:

- 1.- Los Nombres y Apellidos no pueden contener números y caracteres especiales.
- 2.- La fecha de nacimiento deberá ser por lo menos 18 años antes de la fecha actual para poder almacenar un registro, esto debido a que según las leyes de nuestro país para poder hacer cualquier tipo de transacción de carácter legal debe tener como mínimo 18 años de edad para poseer un Documento Único de Identidad (DUI).
- 3.- En la dirección puede utilizar caracteres especiales, sin embargo deberá usarlos teniendo la discreción del caso a fin que la dirección sea entendible cuando se consulte.
- 4.- En los campos DUI y Teléfono solamente serán entendidos como válidos siempre y cuando escriba la cantidad de números requeridos.

En caso que se omita alguna de las directrices anteriores, obtendríamos el siguiente mensaje de error al presionar el botón guardar:

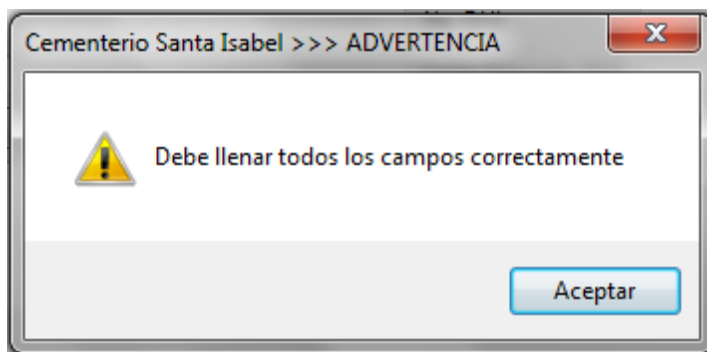


Figura 6.1.31 Mensaje de Error de Nuevo Cliente

B: Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.30 con el botón habilitado y los campos que recopilan los datos del cliente se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran y nos permitirá proporcionar la información para guardar al nuevo cliente, siguiente las directrices del literal anterior.

C: Representa al botón guardar, que deberá presionarse cuando hayamos proporcionado todos los datos necesarios para guardar un nuevo cliente. En caso que no hayamos seguido las directrices antes mencionadas se mostrará en pantalla un mensaje de error tal como lo muestra la Figura 6.1.31.

D: Botón que permite guardar las modificaciones de un registro y que por defecto se encuentra inhabilitado, ya que para que este se encuentre activo primero deberá buscar un cliente (a través del botón buscar, descrito en el literal E) creado anteriormente, seleccionarlo y luego entonces ya que haya hecho las modificaciones que estime necesarias (siempre atendiendo a las directrices recomendadas) este se habilitará para poder guardar los cambios en la información del cliente.

E: Botón que nos permite buscar los clientes que tenemos almacenados en nuestro sistema de información para poder hacer cualquier tipo de modificación, a través de la siguiente Figura 6.1.32.

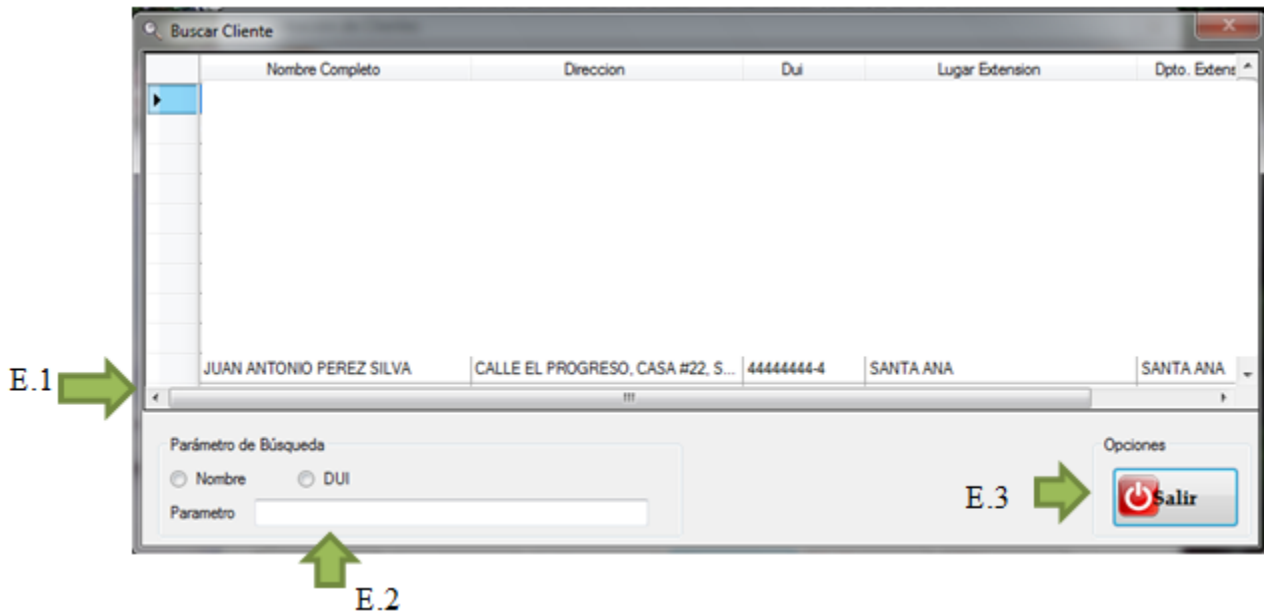


Figura 6.1.32 Búsqueda de Clientes

E.1: Listado de los clientes almacenados en el sistema, para seleccionar uno, solamente debemos hacer clic sobre el registro deseado y nos regresará a la pantalla que muestra la Figura 6.1.30, solamente que con los datos del cliente para proceder a realizar los cambios deseados.

E.2: Campo que nos permite escribir el nombre del cliente que deseamos buscar en caso que el listado de clientes sea muy extenso. Sin embargo antes de eso debemos seleccionar cual será nuestro parámetro de búsqueda en la lista, este puede ser por el número de DUI o por el nombre del cliente. Por lo que solamente necesitamos empezar a escribir el nombre o número deseado para observar que la lista empieza a reducirse filtrando los nombres que contengan los caracteres escritos.

E.3: Botón que nos permite salir de la pantalla de Búsqueda y nos regresa a la pantalla que muestra la Figura 6.1.30.

F: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

G: Botón que nos permite salir de la pantalla que muestra la Figura 6.1.30, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

OPCIÓN PROPIEDAD

A través de este menú podemos acceder a las opciones Nueva propiedad, Beneficiarios y Búsqueda de beneficiarios. Estas opciones nos permiten registrar y administrar las propiedades con que cuenta en venta el Cementerio Santa Isabel para todos aquellos clientes que se encuentran registrados en el Sistema de Información.

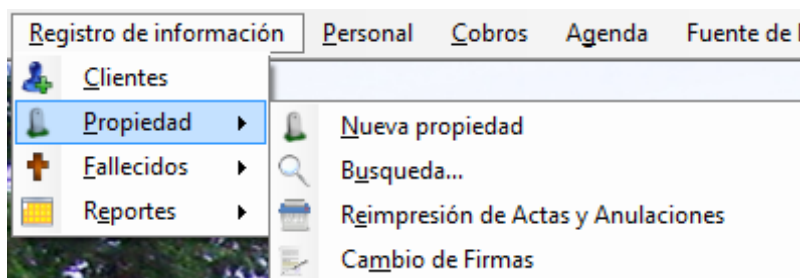


Figura 6.1.33 Submenú Propiedad

Subopción Nueva Propiedad

A través de esta opción que se encuentra dentro de la opción propiedad podemos acceder a la pantalla que nos permite registrar la compra de una propiedad por parte de un cliente registrado con anterioridad. Así mismo automáticamente al guardar la compra de la propiedad se generará el formato del título de propiedad según sea la propiedad que se ha comprado para poder imprimir.

Sin embargo debe recordar que las propiedades no pueden modificarse o eliminarse debido al carácter legal que la impresión de los títulos de propiedad, por lo que antes de guardar se le solicitará confirme si desea realmente guardar el registro de la misma.

Figura 6.1.34 Registro de Propiedades

A: Los datos del comprador son los datos necesarios para poder registrar una compra dentro del sistema de información. Para poder registrarlo correctamente deberá llenar todos los campos solicitados correctamente, siguiente las siguientes directrices:

1.- Deberá dar clic en el botón que se encuentra junto a Codigo Isam, ya que para poder registrar una propiedad esta debe haber sido pagada con sus respectivos impuestos a través de un ISAM, por lo que deberá buscar dicho registro a través de este botón, tal como lo muestra la siguiente figura:

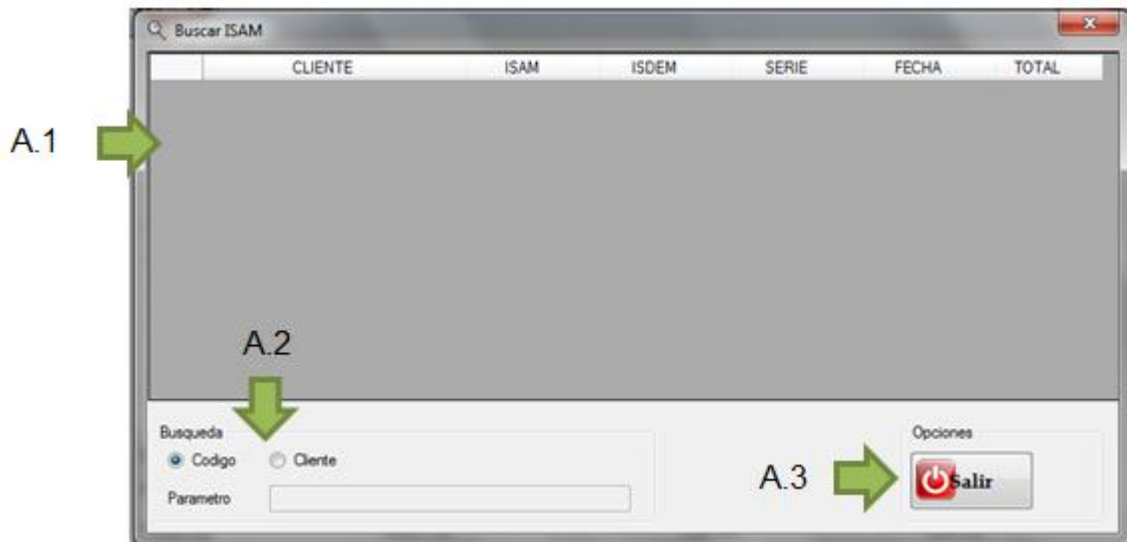


Figura 6.1.35 Buscar ISAM

A.1: Listado de las ISAMS almacenadas en el sistema, para seleccionar uno, solamente debemos hacer clic sobre el registro deseado y nos regresará a la pantalla que muestra la Figura 6.1.34, con el número de ISAM seleccionado, que corresponde al cliente y al compra de la propiedad que se desea asignar al cliente.

A.2: Representan los parámetros de búsqueda dentro de las diferentes ISAMS almacenadas en el sistema, es decir que podemos buscar mediante el código al seleccionar el botón código y escribir el código deseado en el campo parámetro o buscar mediante el nombre del cliente seleccionando el botón cliente y escribiendo su nombre en el campo parámetro, obteniendo el filtro de las coincidencias de la búsqueda automáticamente.

A.3: Corresponde al botón salir que nos permite cerrar esta ventana de búsqueda, haciéndonos regresar a la ventana que muestra la Figura 6.1.34 Registro de Propiedades.

2.- Los Nombres y Apellidos no pueden contener números y caracteres especiales. Sin embargo esta pantalla permite buscar un cliente existente en el sistema y rellenar los campos con los resultados de la búsqueda, tal como muestra la siguiente pantalla de la Figura 6.1.32 Búsqueda de Clientes.

Sin embargo es posible omitir el paso de registrar un nuevo cliente a través de la pantalla provista para esta tarea, al escribir directamente en los campos de esta pantalla la información del cliente, automáticamente el sistema verifica su existencia y si este no existe, procede a guardarlo para su posterior uso.

3.- En la dirección puede utilizar caracteres especiales, sin embargo deberá usarlos teniendo la discreción del caso a fin que la dirección sea entendible cuando se consulte.

4.- En los campos DUI y Teléfono solamente serán entendidos como válidos siempre y cuando escriba la cantidad de números requeridos.

En caso que se omita alguna de las directrices anteriores, obtendríamos el mensaje de error que se muestra en la Figura 6.1.31.

B: En esta area de la pantalla debe introducir la informacion del puesto que desea asignarle al cliente. Esta porcion de la pantalla solamente estará activa si selecciona cualquiera de los dos tipos de puestos (Tipo A o Tipo B), que determinan las medidas del puesto y la cantidad de personas que es posible enterrar en el mismo.

TIPO A		TIPO B	
Ancho	1.20 Mts.	Ancho	3.00 Mts.
Largo	2.70 Mts.	Largo	2.50 Mts.
Cantidad	1-2	Cantidad	1-6

C: En esta area de la pantalla debe introducir la informacion de la sepultura que desea asignarle al cliente. Esta porcion de la pantalla solamente estará activa si selecciona cualquiera de los dos tipos de puestos (Tipo A o Tipo B), que determinan las medidas de la sepultura y la cantidad de personas que es posible enterrar en el mismo.

TIPO A		TIPO B	
Ancho	1.20 Mts.	Ancho	1.10 Mts.
Largo	2.40 Mts.	Largo	2.10 Mts.
Cantidad	1	Cantidad	1

D: Botón que nos permite habilitar los capos principales de la propiedad que por defecto aparecieran inhabilitados, para poder introducir los datos generales de la propiedad a vender al un determinado cliente y dependiendo de la selección de la propiedad se habilitaran los items B o C.

E: Botón que nos permite guardar el registro que hemos digitado para posteriormente generar el titulo de propiedad del tipo que el cliente haya seleccionado según su necesidad. (Ver Figura 6.1.37 Mensaje de Confirmación)

Así mismo al presionar este botón se nos lanzará un formulario para poder introducir los beneficiarios que el dueño de la nueva propiedad designe, tal como lo muestra la Figura 6.1.36 Administración de Beneficiarios:

The screenshot shows a web application window titled "Agregar Beneficiarios". At the top, there are two logos: the national coat of arms of El Salvador on the left and a red circular logo on the right. The main title is "ADMINISTRACIÓN DE BENEFICIARIOS". Below the title is a section labeled "Datos del Beneficiario" containing four input fields: "Cod. Propiedad", "No. DUI", "Nombres", and "Apellidos". Below this section is a large greyed-out rectangular area, likely representing a list of beneficiaries. At the bottom of the window, there is a section labeled "Opciones" with three buttons: "Guardar" (with a floppy disk icon), "Modificar" (with a pencil icon), and "Salir" (with a power button icon). Green arrows labeled E.1, E.2, E.3, E.4, and E.5 point to the input fields, the greyed-out area, and the three buttons respectively.

Figura 6.1.36 Administración de Beneficiarios

E.1: En esta porción de la pantalla debemos introducir los datos que se nos solicitan de cada uno de los beneficiarios que se agreguen a la propiedad (con un mínimo de una persona y un máximo de tres). Por defecto cuando generemos el registro de la compra de una propiedad se lanzara este formulario con el código de la propiedad que se acaba de asignar. Sin embargo si en algún dado momento del tiempo el dueño desea cambiar uno o mas beneficiarios debe hacerse a través de la anulación del título de propiedad y la creación de otro, debido a que esto incurre en un cobro por dicho servicio.

E.2: Muestra el listado de los beneficiarios y sus datos.

E.3: Botón que nos permite guardar los beneficiarios, que deseemos asociarle a la propiedad que pueden ser de 1 a 3 beneficiarios. Hay que aclarar que el botón guardar estará activo siempre y cuando esta pantalla haya sido lanzada al guardar la propiedad, si es accedida a través del menú principal entonces debera seleccionar una propiedad y luego el beneficiario que desea modificar.

E.4: Botón que nos permite modificar uno o varios beneficiarios asociados a la propiedad, este botón solamente estará activo cuando la pantalla sea abierta desde el menú principal.

E.5: Botón que nos permite salir de la pantalla, lo que nos permitirá guardar por completo nuestra propiedad y luego serán abiertas en nuestra pantalla el título de propiedad y actas, según sea el caso de la propiedad comprada.

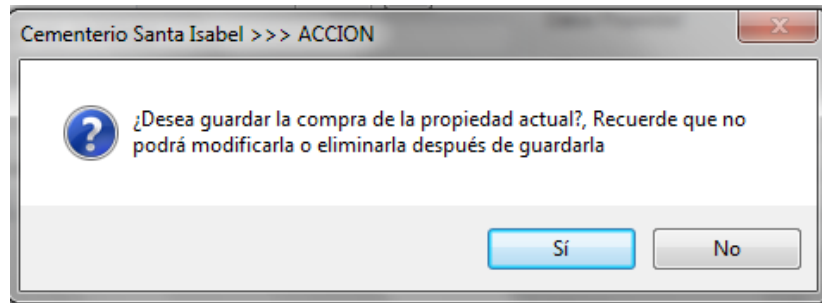


Figura 6.1.37 Mensaje de Confirmación

F: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

G: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

H: Espacio para proporcionar al sistema los datos generales de la propiedad, para la cual se está realizando la venta.

Subopción Búsqueda

Esta opción nos permite buscar las diferentes propiedades vendidas a través del nombre del dueño, los beneficiarios de las propiedades, asentamiento de partida de defunción y nombre del fallecido, es decir relacionado a las propiedades que existen en la colección de datos almacenados.

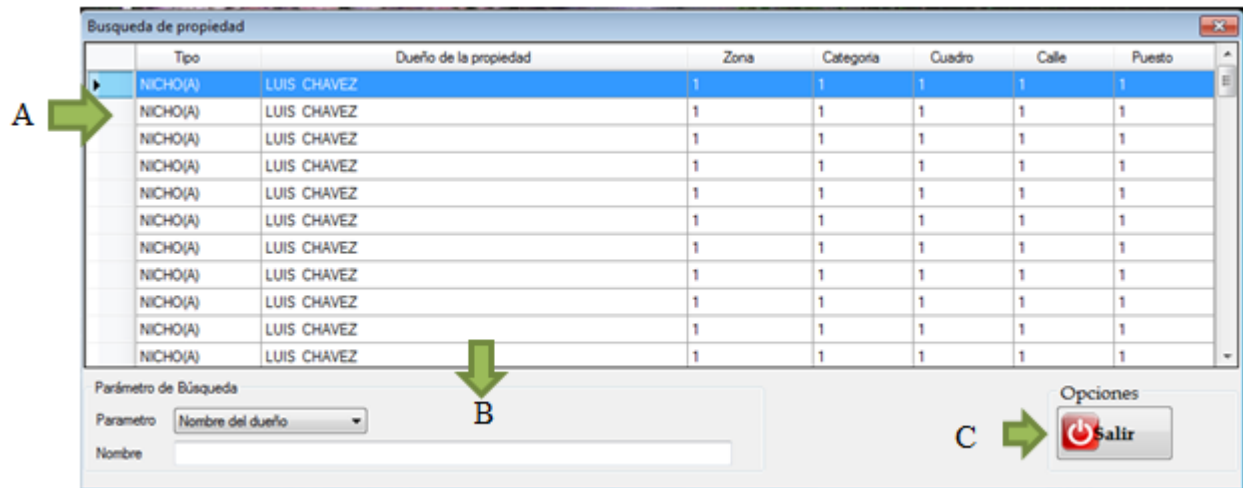


Figura 6.1.38 Búsqueda de Propiedades

- A:** Listado de las propiedades vendidas por el cementerio.
- B:** Selección de parámetros para la búsqueda de propiedades, fallecidos y beneficiarios. Al seleccionar el parámetro de búsqueda se debe escribir un texto relacionado al parametro de búsqueda seleccionado para filtrar la informacion que se muestra en la lista.
- C:** Botón que nos permite salir de el formulario de búsqueda.

Subopción Reimpresión de Actas y Anulaciones

A través de este menú el sistema nos proporciona la oportunidad de anular títulos y actas de propiedad, para poder hacer cambios en los títulos de propiedad, como por ejemplo si un cliente quisiera cambiar los beneficiarios de sus propiedad debe utilizar esta opción para anular el titulo actual y luego generar un titulo nuevo, tal como vimos en la sección Submenú Propiedades. Dentro de esta opción podremos observar la siguiente pantalla:

Figura 6.1.39 Reimpresión de Actas y Anulación de Títulos

A: Muestra los datos del comprador de la propiedad, sin embargo en este formulario para obtener toda esta información primero debe seleccionar el tipo de propiedad, luego escribir el código del título de la propiedad, presionamos el botón buscar que se encuentra junto a la caja de texto del código del título. Al realizar este proceso entonces obtendremos toda la información de dicho título, en caso que este exista.

B: Información de la ubicación del puesto en caso que la propiedad sea un nicho, en caso contrario solamente observara “0” en todas las cajas de texto.

C: Datos generales de la propiedad de la propiedad que se buscó para anular o reimprimir el acta de propiedad en calidad de reposición.

D: Información de la ubicación del puesto en caso que la propiedad sea una fosa, en caso contrario solamente observara “0” en todas las cajas de texto.

E: Botón que nos permite anular el título de propiedad buscado, al presionar el botón se nos mostrará un formulario en el que debemos introducir un usuario y su contraseña con los permisos adecuados para poder autorizar la nulidad del titulo de propiedad, esto debido a que es una transacción de carácter legal. (Tal como lo muestra la Figura 6.1.40 Autorización de Anulación y Figura 6.1.41 Jusstificación de Anulación)

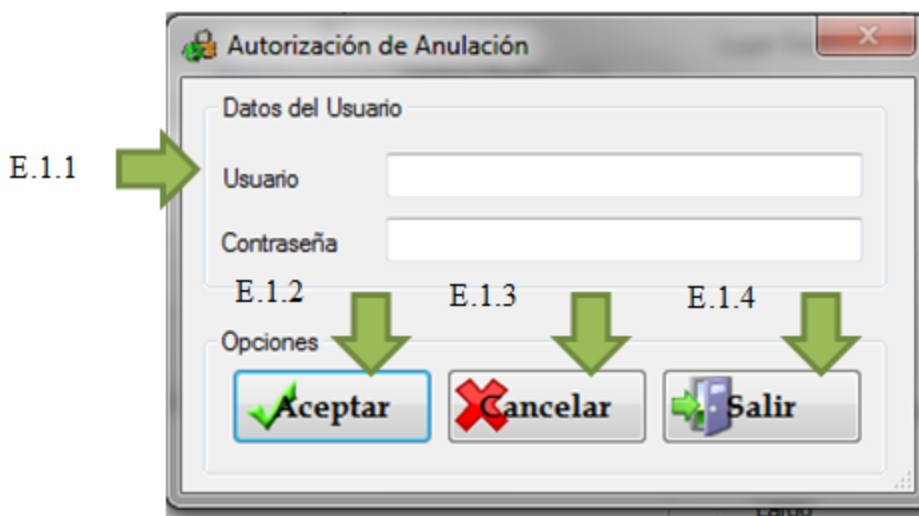


Figura 6.1.40 Autorización de Anulación

E.1.1: Datos del usuario, en el espacio provisto con el nombre Usuario, deberá introducir un usuario con permisos de administrador para poder autorizar la anulación del titulo de propiedad. Asi mismo deberá introducir la respectiva clave del usuario para validar dicho usuario y la autorizar la anulación. Si el usuario introducido cuenta con los permisos para realizar esta actividad, deberá observar la Figura 6.1.26 Jusstificación de Anulación.

E.1.2: Botón que nos permite validar el usuario y contraseña proporcionado al interfaz que captura los datos.

E.1.3: Botón que nos permite cancelar la acción de validación del usuario digitado en los espacios del formulario, por lo que después de presionar dicho botón las cajas de texto será limpiada para introducir un nuevo usuario.

E.1.4: Botón que nos permite salir del formulario, regresándonos al formulario mostrado en la Figura 6.1.39 Reimpresión de Actas y Anulación de Titulos, sin embargo recuerde que esto implica que no se anuló el titulo de propiedad. Así mismo si no cuenta con un usuario valido tampoco se hará efectiva la anulación.

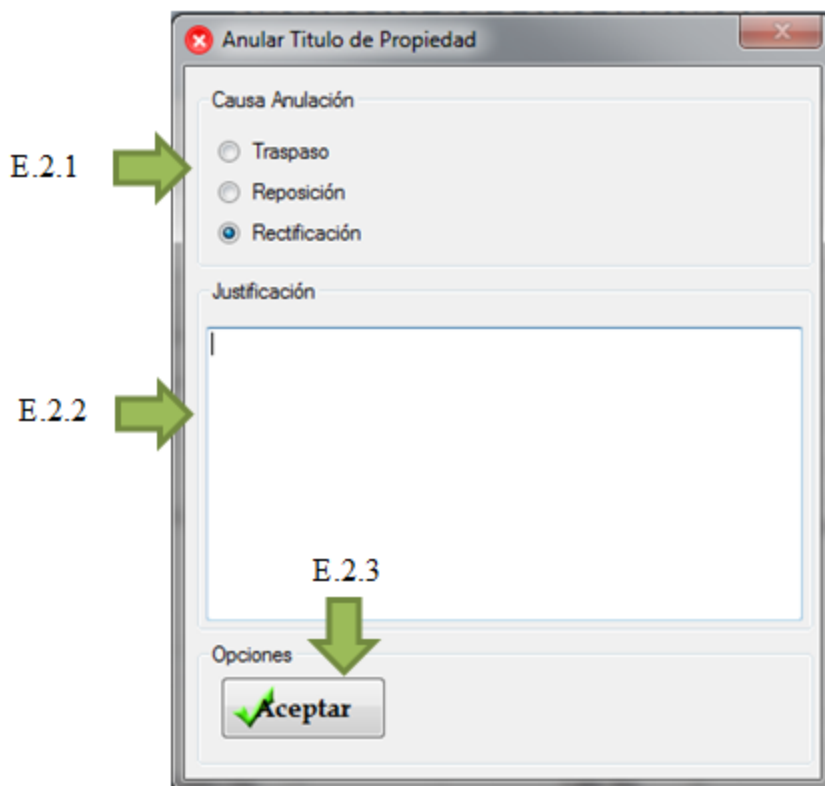


Figura 6.1.41 Justificación de Anulación

E.2.1: Área en donde debe seleccionar la causa de anulación del título de propiedad

E.2.2: Área en la que se especifica el motivo por el cual se anulará el título a propiedad.

E.2.3: Botón que permite confirmar el almacenamiento de la justificación de la anulación del título a propiedad.

F: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

G: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

H: Botón que nos permite imprimir el acta de propiedad de un nicho o fosa, según sea el caso de la propiedad obtenida en la búsqueda. Esta se mostrará tal como se muestra en la Figura 6.1.42 Impresión de Actas.

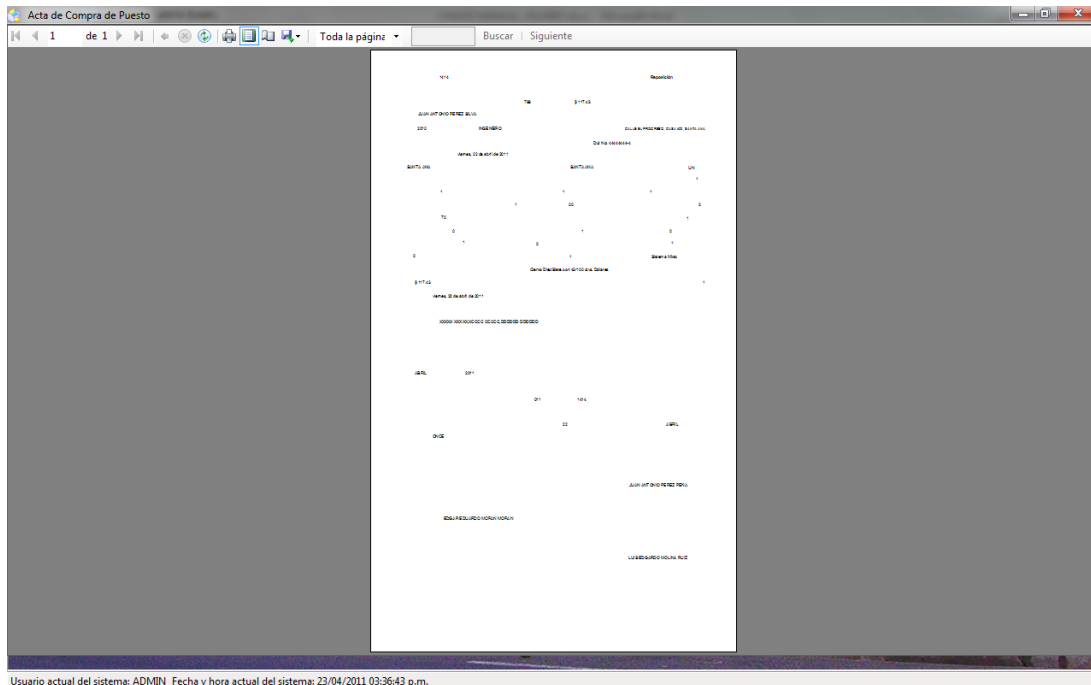


Figura 6.1.42 Impresión de Actas

Subopción Cambio de Firmas

Esta opción del menú nos permite cambiar las firmas que aparecen impresas en los títulos de propiedad y actas, ya que en un periodo determinado pueden cambiar las autoridades, por lo que el sistema está provisto para registrar esos cambios.



Figura 6.1.43 Cambio de Firmas

A: Campos que permiten mostrar y almacenar los nombres de las autoridades que firman las actas y títulos de propiedad.

B: Botón habilitar las cajas de texto en que se muestran los nombres de las autoridades para poder realizar cualquier tipo de cambio.

C: Botón que nos permite guardar cualquier modificación hecha a los nombres de las autoridades actuales.

D: Botón que nos permite cancelar cualquier proceso de edición que realicemos a los nombres de las autoridades.

E: Botón que nos permite salir del formulario actual.

OPCIÓN FALLECIDOS

A través de este menú podemos acceder a las opciones Partida de defunción y registro de inhumaciones y exhumaciones para todos aquellos clientes que se encuentran registrados en el Sistema de Información.

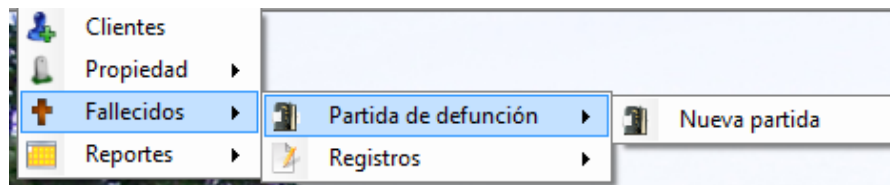


Figura 6.1.44 Submenú Fallecidos

Subopción Partida de defunción

A través de esta opción se pueden registrar en el Sistema de Información las partidas de defunción de las personas fallecidas en el departamento de Santa Ana.

Figura 6.1.45 Creación de partidas de defunción

A: En esta área se deberán escribir el numero de la partida de defuncion y el nombre del solicitante de la misma, ademas de la busqueda del numero de la ISAM que será asociada a la compra de dicha partida, tal como lo muestra la Figura 6.1.20 Buscar ISAM.

B: En esta área de la pantalla deberá escribir los datos de la persona fallecida, evitando dejar campos vacios, ya que si alguno de los campos se encuentra vacio no se nos permitirá guardar la partida de defunción mostrando el mensaje mostrado en la pantalla de la Figura 6.1.31 Mensaje de Error de Nuevo Cliente.

Cuando se selecciona un fallecimiento en casa entonces al presionar el botón guardar se nos lanzara el formulario que muestra la Figura 6.1.46 Agregar Testigos de Defunciones.

The screenshot shows a web application window titled "Agregar Testigos". The main heading is "TESTIGOS DE DEFUNCIONES". Below this, there is a section "Datos del Testigo" with the following fields: "No. Partida Defunción" (value: 3333), "Nombres", "Apellidos", "No. DUI", and "Parentesco". Below the form is a table with columns "Nombres", "Apellidos", "Parentesco", and "Dui". Below the table are five buttons labeled "B.3" through "B.7", which correspond to "Nuevo", "Agregar", "Guardar", "Cancelar", and "Eliminar" respectively. The "Agregar" button has a green plus sign, and the "Cancelar" button has a red X.

Figura 6.1.46 Agregar Testigos de Defunciones

B.1: Espacio del formulario en el que debe introducir la información de los testigos del fallecimiento en casa (deben ser tres testigos) que avalen la muerte.

B.2: Lista de los testigos que se introducen.

B.3: Botón que nos permite generar un nuevo conjunto de testigos para una defuncion. Este boton quedará deshabilitado una vez que se presione por primera vez y

nos habilitará los campos del literal B.1 para poder introducir la información de los diferentes testigos.

B.4: Botón que nos permite agregar cada uno de los testigos que desean registrar y se deshabilitará al introducir el tercer testigo de la defunción.

B.5: Botón que nos permite guardar los testigos de la defunción asociados al número de partida de defunción que se muestra en el literal de B.1.

B.6: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

B.7: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

C: Botón que nos permite generar una nueva partida de defunción, habilitando los campos necesarios para registrarla.

D: Botón que nos permite guardar la partida de defunción en caso que toda la información proporcionada tenga el formato correcto. Al presionar este botón se nos preguntará si estamos seguros de guardar el registro debido a que por el carácter legal del documento este no puede modificarse o eliminarse.

E: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

F: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

Subopción Registros

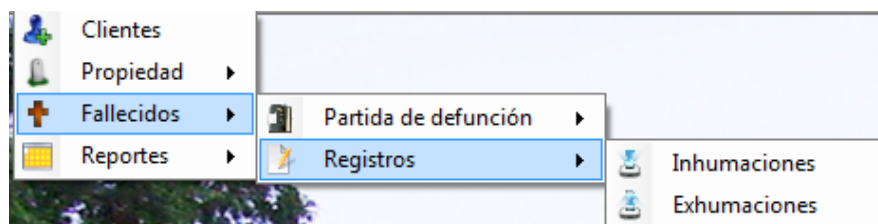


Figura 6.1.47 Submenú Registros

A través de este menú podemos acceder a las opciones de registro de inhumaciones y exhumaciones. Para poder registrar una inhumación primero se debe haber registrado una partida de defunción y para registrar una exhumación primero debe haberse registrado con anterioridad una inhumación.



Figura 6.1.48 Registrar Inhumaciones

A: Área provista para obtener toda la información necesaria para poder registrar una inhumación dentro del sistema. Debemos aclarar que se pueden observar botones con una lupa, lo que significa una búsqueda, en el caso de la propiedad y el código del ISAM estos ya fueron explicados con anterioridad en la 6.1.38 Búsqueda de Propiedad y Figura 6.1.35 Buscar ISAM. Por lo que solamente explicaremos la pantalla de búsqueda de las partidas de defunción en la Figura 6.1.49 Buscar Partida de defunción.

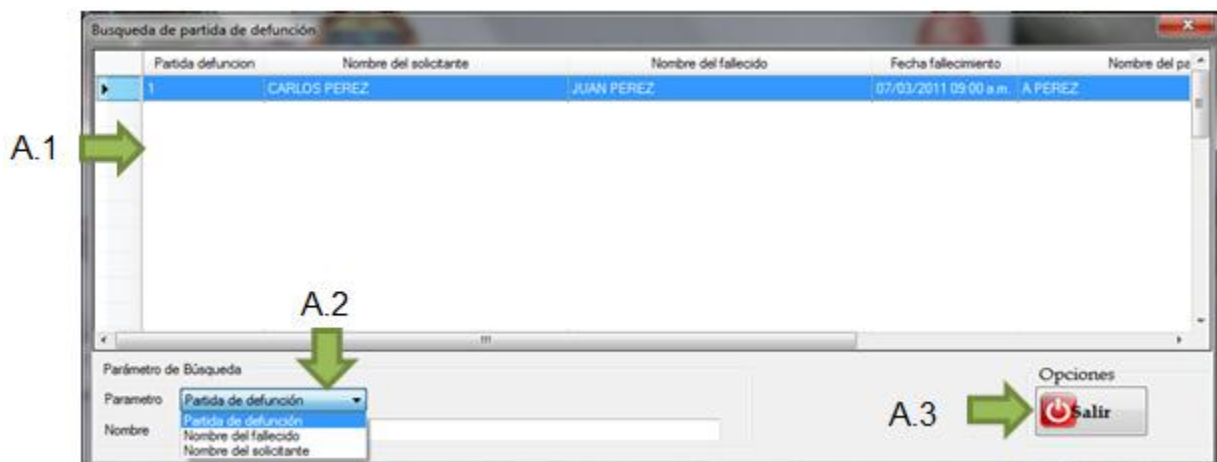


Figura 6.1.49 Buscar Partida de defunción

A.1: Listado de partidas de defunción almacenadas en el sistema.

A.2: Parámetros de búsqueda, que nos permite seleccionar el tipo de busque que deseamos hacer para encontrar una determinada partida de defunción, entre las opciones tenemos la búsqueda por número de partida, nombre del fallecido o nombre del solicitante, lo que posteriormente escribiremos en el campo de texto de esta misma área.

A.3: Botón que nos permite salir de esta pantalla y regresarnos a la pantalla mostrada en la Figura 6.1.48 Registrar Inhumaciones.

B: Nos permite generar una nueva inhumación, ya que por defecto esta pantalla tendrá deshabilitados los campos para escribir los datos de la misma, por lo que al presionar este botón se habilitaran para poder recibir la información.

C: Botón que nos permite guardar la inhumación, al tener todos los datos completos y en el formato adecuado según lo solicita la pantalla. Recuerde que debe ser breve cuando rellene los campos del formulario, ya que esta información será impresa en los formularios que posee el cementerio para tal fin.

D: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

E: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

Figura 6.1.50 Registro de Exhumación

A: área provista para proporcionar todos los datos necesarios para registrar en el sistema de información una exhumación, por lo que deberá seguir las normas planteadas para un buen formato en los campos que se le solicitan. Así mismo en esta área deberá buscar un registro de inhumación para poder guardar el registro, ya que no se puede registrar una exhumación si en una fecha anterior no se registró una exhumación de una determinada persona. Esto lo haremos a través de la Figura 6.1.48 Registro de Inhumaciones.

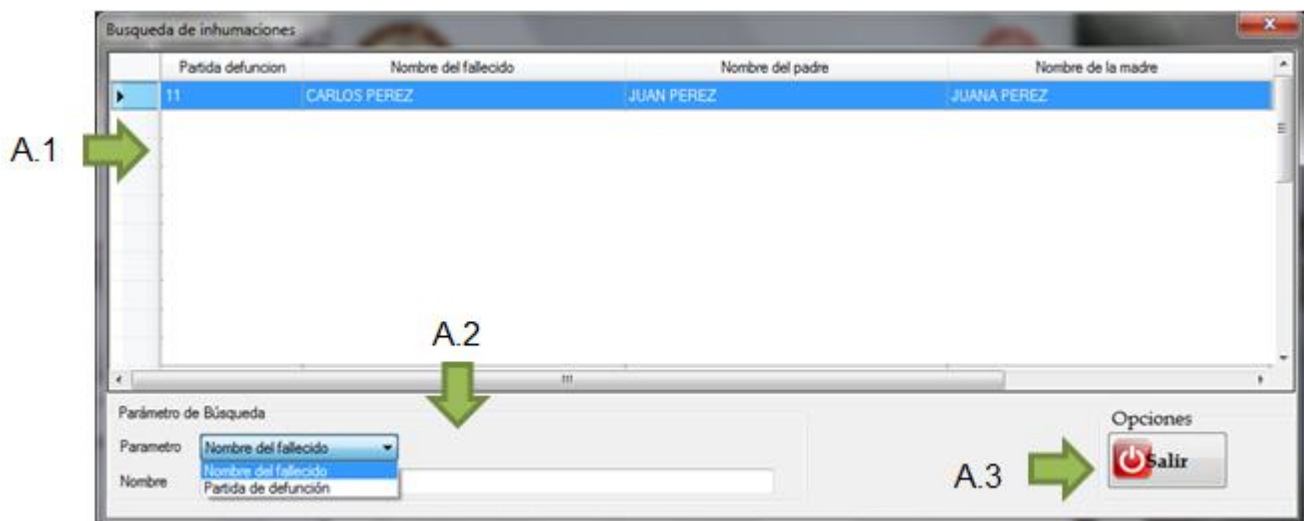


Figura 6.1.51 Búsqueda de Inhumaciones

A.1: Listado de las inhumaciones registradas por el sistema de información.

A.2: Parámetros de búsqueda para encontrar una determinada inhumación, a través del nombre del fallecido o por el número de la partida de defunción. Esto se logrará seleccionando el parámetro por el cual se desea buscar y luego escribirlo en la caja de texto. Cuando se haga esto verá que automáticamente la información se ha filtrado según los caracteres de coincidencia que escribió.

A.3: Botón que nos permite salir de esta pantalla y nos llevará de nuevo a la pantalla de la Figura 6.1.50 Registro de Exhumación.

B: Nos permite generar una nueva exhumación, ya que por defecto esta pantalla tendrá deshabilitados los campos para escribir los datos de la misma, por lo que al presionar este botón se habilitaran para poder recibir la información.

C: Botón que nos permite guardar la exhumación, al tener todos los datos completos y en el formato adecuado según lo solicita la pantalla. Recuerde que debe ser breve cuando rellene los campos del formulario, ya que esta información será impresa en los formularios que posee el cementerio para tal fin.

D: Botón que nos permite cancelar cualquier acción que nos encontremos realizando.

E: Botón que nos permite salir de la pantalla, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán.

Opción Reportes

A través de este submenú podremos acceder a dos reportes elaborados para el área del registro de la información, Exhumaciones, que es un listado de todas la exhumaciones que se han llevado a cabo durante un periodo determinado de tiempo y el reporte servicios generales, que es un resumen que nos muestra la cantidad de servicios brindados a la población durante un periodo determinado de tiempo.

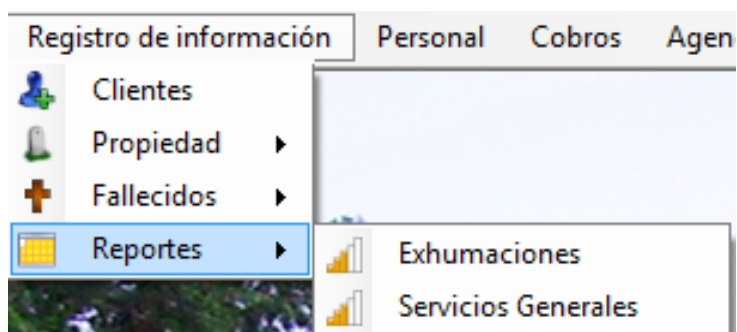


Figura 6.1.51 Submenú Reportes

Subopcion Exhumaciones

Este reporte le permitirá consultar las exhumaciones registradas durante un período determinado de tiempo, tal como muestra la Figura 6.1.52 Reporte de Control de Exhumaciones.

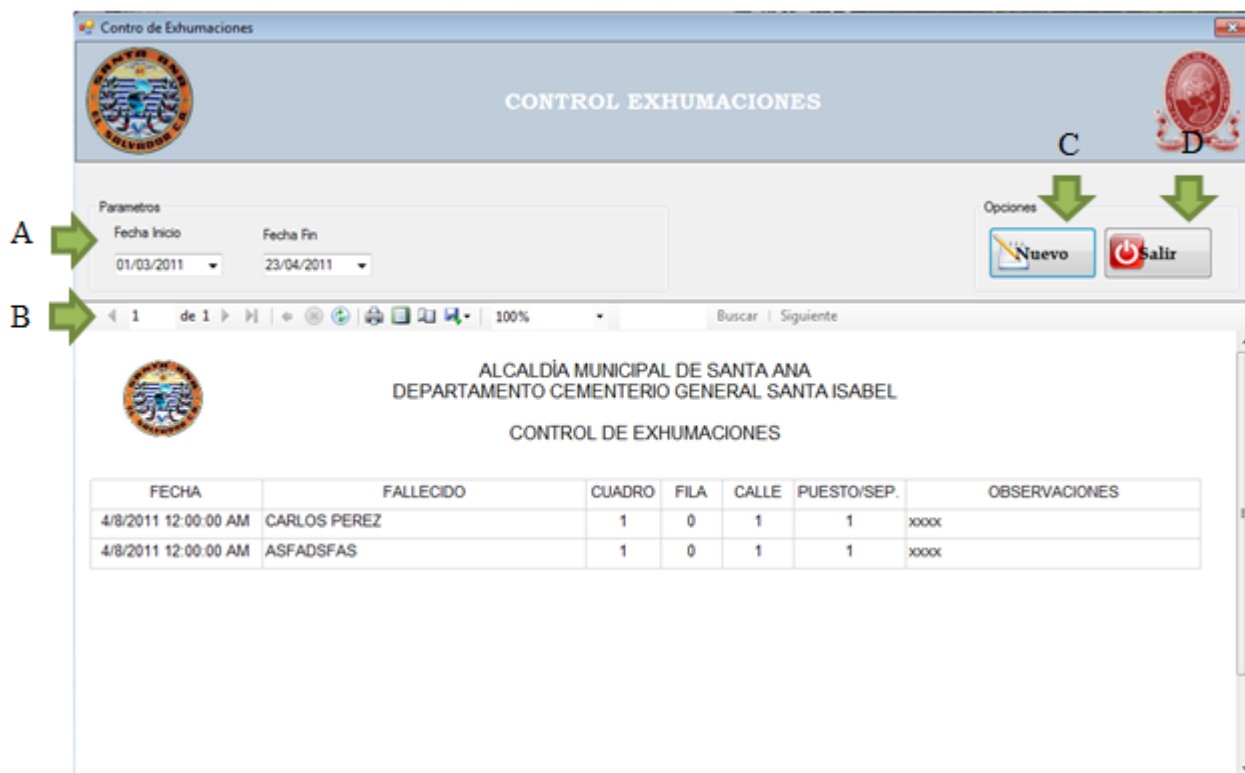


Figura 6.1.52 Reporte Control de Exhumaciones

A: Parámetros de búsqueda de la información, que nos permite seleccionar un rango de fechas en las cuales queremos que se nos presente la información de las exhumaciones.

B: Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

C: Botón que nos permite generar un nuevo reporte con los parámetros seleccionados en el literal "A".

D: Botón que nos permite salir del formulario.

Subopcion Servicios Generales

Este reporte nos permitirá obtener la cantidad de propiedades vendidas durante el periodo de tiempo deseado, así como la cantidad de defunciones, inhumaciones y exhumaciones que ha proporcionado el cementerio.



Figura 6.1.53 Reporte Estadístico

A: Parámetros de búsqueda de la información, que nos permite seleccionar un rango de fechas en las cuales queremos que se nos presente la información de los servicios brindados a la población.

B: Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

C: Botón que nos permite generar un nuevo reporte con los parámetros seleccionados en el literal “A”.

D: Botón que nos permite salir del formulario.

Opción Fuente de Datos

Esta opción nos permite respaldar la información que existe en el sistema, utilizando cada una de las tablas que almacenan información. Así mismo nos permite cambiar la fuente de datos a la que se conecta el sistema de Información.

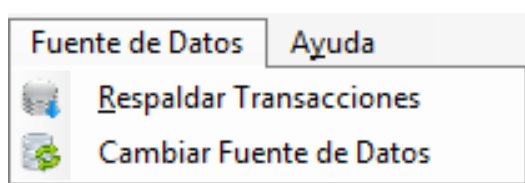


Figura 6.1.54 Submenú Fuente de Datos

Subopción Respalda Transacciones

Esta opción nos permite respaldar la información de nuestra base de datos, debe recordar que para ello deberá seleccionar cada una de las tablas que desea respaldar. Al realizarse el respaldo podrá encontrar una carpeta en el C: del equipo denominada “RespaldoCementerio” donde podrá encontrar el respaldo en un archivo de Microsoft Excel. Debe tomar en cuenta que esta información será solamente de lectura y no podrá modificarse sin la autorización pertinente.



Figura 6.1.55 Respalda Transacciones

- A:** Lista de las tablas que contiene nuestra base de datos
- B:** Botón que nos permite generar el respaldo de la información seleccionada. Si todo se realizó satisfactoriamente se nos lanzara una pantalla como la de la Figura 6.1.56.

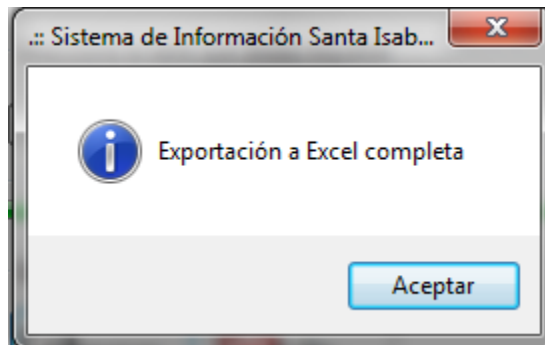


Figura 6.1.56 Respaldo Transacciones

- C:** Botón que nos permite salir del formulario actual.

Subopción Cambiar Fuente de Datos

A través de esta opción podemos cambiar la cadena de conexión del servidor de base de datos que utiliza el sistema de información. Por defecto se tendrá una cadena de conexión por defecto y si cambia esta cadena recuerde que debe hacerse por un nombre de servidor valido, ya que si este no lo es no podrá utilizar el sistema de información.



Figura 6.1.57 Cambiar Fuente de Datos

A: Cadena de conexión que utiliza el sistema de información para obtener y guardar toda la información.

B: Botón que nos permite editar la cadena de conexión.

C: Botón que nos permite guardar los cambios en la cadena de conexión.

D: Botón que nos permite cancelar cualquier cambio que hayamos realizado a la cadena de conexión, siempre y cuando se haya presionado el botón guardar.

E: Botón que nos permite salir del formulario actual.

MENÚ PERSONAL

Permite administrar todo lo relacionado al personal de cementerio, incluyendo a los contratistas que son personas externas al cementerio.

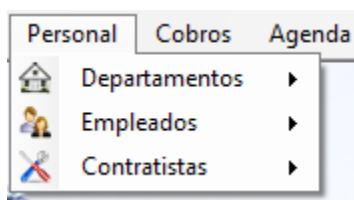


Figura 6.1.58 Personal

INGRESO A SECCIÓN DE DEPARTAMENTOS

Esta sección permitirá agregar departamentos así como modificarlos alguno que se halla ingresado y necesite cambios, ya sea de nombre o corrección ortográfica.

Para ingresar al módulo a la sección de departamentos es necesario ir al menú departamentos y luego hacer clic sobre la opción Agregar/Modificar tal como se muestra en la figura 6.1.59



Figura 6.1.59 Personal

Formulario Agregar/Modificar departamentos



Figura 6.1.60

- A. Lugar en el cual se debe ingresar el departamento
- B. Habilita la caja de texto y el botón guardar para realizar un inserción.
- C. Permite guardar el departamento
- D. Se utiliza para modificar un departamento luego de haber hecho clic sobre un dato.
- E. Cancela el proceso de inserción o modificación.
- F. Permite salir de la ventana.
- G. Muestra los departamentos ingresados al sistema.

Modulo de personal

Este modulo permite agregar, modificar empleados, agregar vacaciones, y horarios de trabajo. A demás permite agregar y modificar contratistas.

A continuación se muestra en la figura 6.1.61 el menú para dicho modulo.

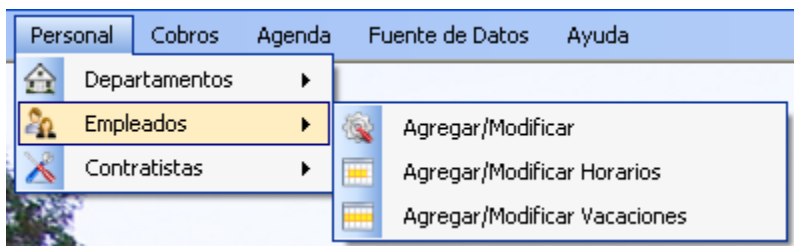


Figura 6.1.61

En la figura 6.1.62 se muestra la pantalla para agregar un empleado

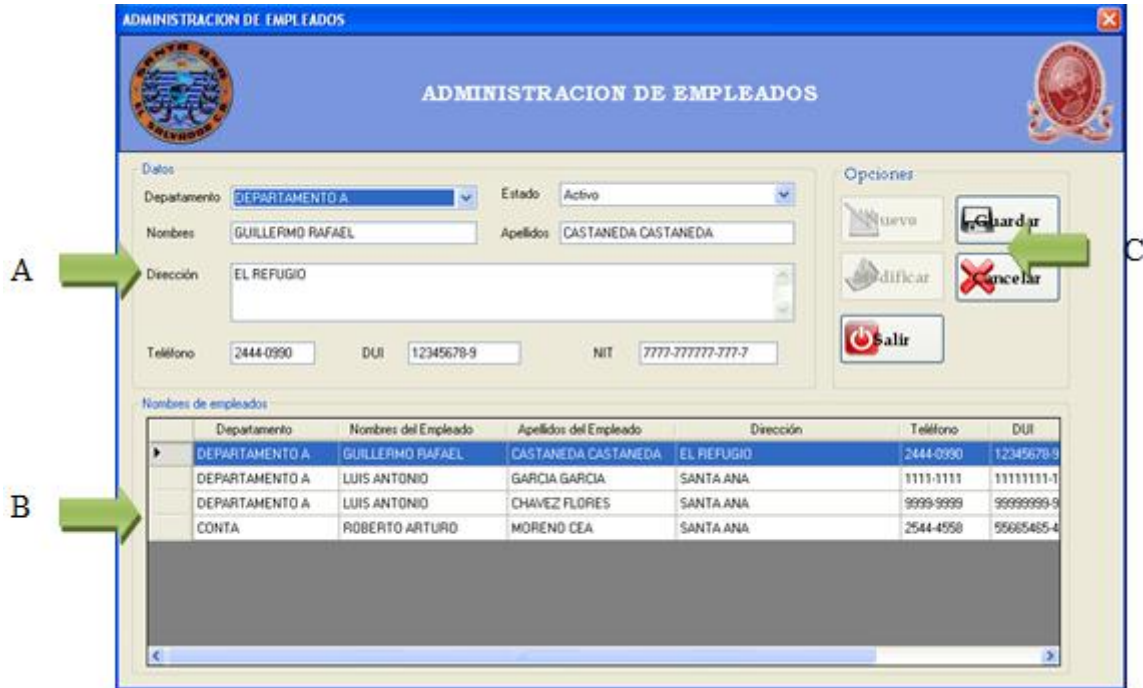


Figura 6.1.62

- A. En esta sección aparecen los datos de identidad personal del empleado, por ejemplo, DUI, NIT, dirección, teléfono entre otras.
- B. Muestra los empleados que ya han sido ingresados al sistema.
- C. Este bloque contiene los botones de control los cuales se utilizan para agregar, modificar empleados, cancelar operaciones y salir de la ventana.

Horario de Labores

En esta sección se permite asignar un horario a cada empleado, pudiendo especificar el día y la cantidad de horas que este prestara sus servicios Dicho lugar se muestra en la figura 6.1.63



Figura 6.1.63

- A. Muestra todos los empleados ingresados al sistema, para agregar horas y días a un empleado basta con hacer clic sobre uno de ellos.
- B. Muestra los días y cantidad de horas asignadas a un empleado.
- C. Permite indicar especificar el día que se le asignara a un empleado.
- D. Botones de control ya sea para guardar o modificar un dato.

Asignación de vacaciones

Permite asignar vacaciones a un empleado, asignando tanto la fecha de inicio como la fecha de finalización de la misma.

En la figura 6.1.64 se muestra la pantalla en la cual es posible asignar vacaciones a los empleados.



Figura 6.1.64

- A. Muestra los empleados ingresados a los cuales se les podrá asignar vacaciones.
- B. Muestra el inicio y finalización del periodo de vacaciones del empleado seleccionado.
- C. Permite especificar la fecha de inicio y fin del periodo.
- D. Botones de control para agregar, modificar y cancelar operaciones.

Contratistas

Permite agregar y modificar contratistas.

En la figura 6.1.65 se muestra la pantalla para dicho proceso.

The screenshot shows a web application window titled "ADMINISTRACION DE CONTRATISTAS". The window has a blue header with the title and two logos. Below the header, there are three main sections:

- Datos personales:** A form with input fields for "Nombres", "Apellidos", "Direccion", "Teléfono", "DUI", and "NIT". A dropdown menu for "Estado" is set to "Activo".
- Opciones:** A vertical stack of buttons: "Nuevo", "Guardar", "Modificar", "Cancelar", and "Salir".
- Contratistas:** A table with columns for "Nombres del contratista", "Apellidos del contratista", "Dirección", "Teléfono", and "DUI". The table is currently empty.

Three green arrows point to specific elements: Arrow A points to the "Teléfono" field; Arrow B points to the "Contratistas" table; Arrow C points to the "Modificar" button.

Figura 6.1.65

- A. Permite ingresar información personal del contratista.
- B. Muestra los contratistas ingresados al sistema.
- C. Botones de control.

MENÚ COBROS

Permite administrar todo lo relacionado los cobros que hace el cementerio, esto incluye la generación y anulación de la factura ISAM, agregar y modificar servicios e impuestos.

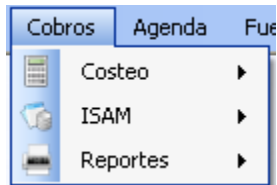


Figura 6.1.66 Cobros

ADMINISTRACIÓN DE SERVICIOS E IMPUESTOS

Para ingresar a la sección de agregar y modificar servicios es necesario hacer clic en el menú Cobros, opción costeo y Agregar/Modificar Servicios tal como se muestra en la figura 6.1.67

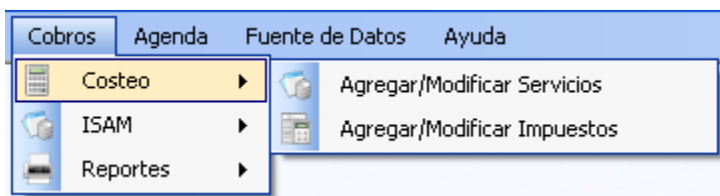


Figura 6.1.67

Cabe mencionar que para los impuestos es el mismo procedimiento la única diferencia es que ahora es necesario hacer clic sobre la opción Agregar/Modificar Impuestos.

CAPTURA DE SERVICIOS (Figura 6.1.68)

Nombre del servicio	Descripción del servicio	Tipo	Valor
INHUMACION	ENTERRAR UN CUERPO	Fijo	14.00
EXHUMACION	ESTRAER UN CUERPO	Fijo	14.00

Figura 6.1.68

- A. Permite agregar el nombre del servicio.
- B. En esa sección es posible agregar una descripción del servicio que se está agregando.
- C. Permite especificar si el servicio que se está ingresando posee un valor fijo o es calculado en base a servicios realizados a la institución.
- D. Permite especificar el costo del servicio.
- E. Muestra todos los servicios ingresados al sistema.
- F. Se utiliza para indicar al sistema que se ingresara un nuevo servicio.
- G. Se utiliza para agregar un nuevo servicio, basta simplemente con llenar todos los campos y luego hacer clic sobre dicho botón.
- H. Se utiliza para modificar un registro previamente ingresado, basta simplemente con hacer clic sobre un servicio ingresado en la tabla y este se cargara en todos los campos.

I. Se utiliza para salir de la pantalla de servicios.

PANTALLA DE IMPUESTOS (*Figura 6.1.69*)

Nombre del impuesto	Descripción del impuesto	Valor
FONDO FIESTA	FONDO PARA FESTEJOS MUNICIPALES	0.03

Figura 6.1.69

En esta sección es donde se agregan los impuestos que se manejan a nivel de cementerio, dicha pantalla se muestra en la figura 6.1.69 la cual con cual consta de las siguientes partes:

- A. En este campo es donde se ingresa el nombre del impuesto.
- B. Corresponde a la descripción del impuesto a agregar.
- C. Se debe especificar el valor del impuesto en porcentaje.
- D. Muestra los impuestos agregados al sistema.
- E. Habilita los botones y cajas de texto para ingresar un nuevo impuesto.
- F. Permite guardar toda la información ingresada al sistema, es de aclarar que esta debe estar completa en su totalidad.

- G. Permite modificar un impuesto previamente agregado, para ello es necesario hacer un clic sobre el impuesto mostrado en la grafica, luego de haber hecho clic sobre el impuesto se cargara toda la información en los campos correspondientes para editar la misma. Una vez realizados los cambios es necesario hacer clic sobre este botón para confirmar la modificación. Luego de haber hecho clic sobre el botón modificar se le preguntara si está seguro de aplicar los cambios, dicho mensaje se muestra en la figura 6.1.70 y por último el sistema le informara que el impuesto se modifico de forma correcta según lo que se muestra en la figura 6.1.71
- H. Permite cerrar la sección de impuestos.

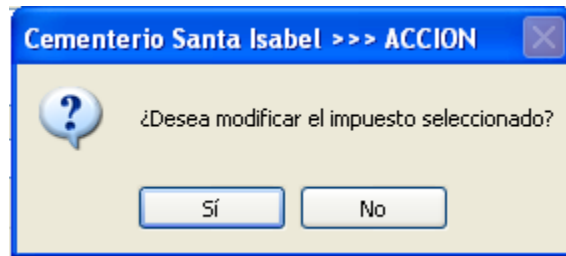


Figura 6.1.70



Figura 6.1.71

SECCION DE FACTURA ISAM (Figura 6.1.72 Cobros)

En la figura 6.1.72 se muestra el menú para ingresar a la sección donde se emiten las facturas ISAM.

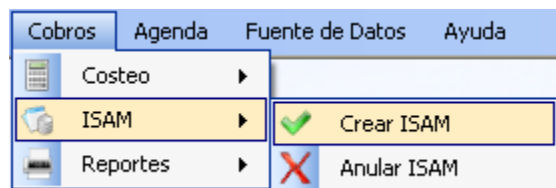


Figura 6.1.72

A continuación se muestra el formulario en el cual se ingresan las facturas ISAM, dicho formulario se muestra en la figura 6.1.73

Figura 6.1.73

- A. Corresponde al número de ISDEM
- B. Corresponde al número de ISAM
- C. Se debe ingresar la serie de la ISAM
- D. Muestra la fecha y hora de creación de la factura
- E. Muestra a nombre de quien se emitirá la factura
- F. Muestra el servicios que se cobrara
- G. Muestra el impuesto que se cargara al servicio antes mencionado
- H. Muestra el total de la factura
- I. Muestra todos los servicios e impuestos agregados, visualizando así la cantidad de servicios, el tipo ya sea servicio o impuesto, concepto o sea el nombre del

impuesto o servicio que se cobra y el costo individual de cada uno de los ítems agregados.

- J. Permite seleccionar un cliente, es decir, al hacer clic sobre este botón carga otra pantalla en la cual se podrá buscar y seleccionar un cliente Fig 6.1.75
- K. Carga otra pantalla que contiene toda la lista de servicios que brinda el cementerio. Figura 6.1.76
- L. Muestra otra pantalla que contiene toda la lista de impuestos ingresados. Fig 6.1.77
- M. Al presionar este botón se cargaran a la factura tanto el impuesto como el servicio que selecciono, cabe mencionar que para ello deben de estar completos todos los campos anteriores de lo contrario el sistema le pedirá que ingrese toda la información requerida o le informara que el servicio ya está ingresado, es por ello que se pide la cantidad de servicios que se ingresaran.
- N. Al cargar el formulario debe hacer clic sobre este botón para habilitar todos los botones y cajas de texto para ingresar información.
- O. Con este botón agregara al sistema la ISAM una vez que se hallan llenado todos los campos requeridos, así como también al hacer clic sobre dicho botón se generara la impresión de la factura actual.
- P. Con este botón permite limpiar el formulario y llenar de nuevo la factura.
- Q. Este botón permite cerrar la ventana y volver al menú principal.

A continuación en la figura 6.1.74 se muestra una captura de una factura completamente llena lista para guardar e imprimir.

Cantidad	Tipo	Concepto	Precio Unitario	Detalle
3	SERVICIO	INHUMACION	\$ 14.00	\$ 42.00
1	IMPUESTO	FONDO FIESTA	\$ 00.42	\$ 01.26
2	SERVICIO	EXHUMACION	\$ 07.00	\$ 14.00
1	IMPUESTO	FONDO FIESTA	\$ 00.21	\$ 00.42

Figura 6.1.74

BUSQUEDA DE CLIENTES PARA AGREGAR A LA FACTURA ISAM

Para este fin se cuenta con un pantalla en el cual se muestran todos los clientes previamente ingresados al sistema en el cual se podrá seleccionar a uno de ellos al hacer clic sobre el o al escribir ya sea su DUI o su nombre.

Dicha pantalla permite hacer búsquedas ya sea por nombre o por DUI simplemente es de especificar con cual se realizara la búsqueda y escribir la información correspondiente. En la figura 6.1.75 se muestra dicha pantalla.

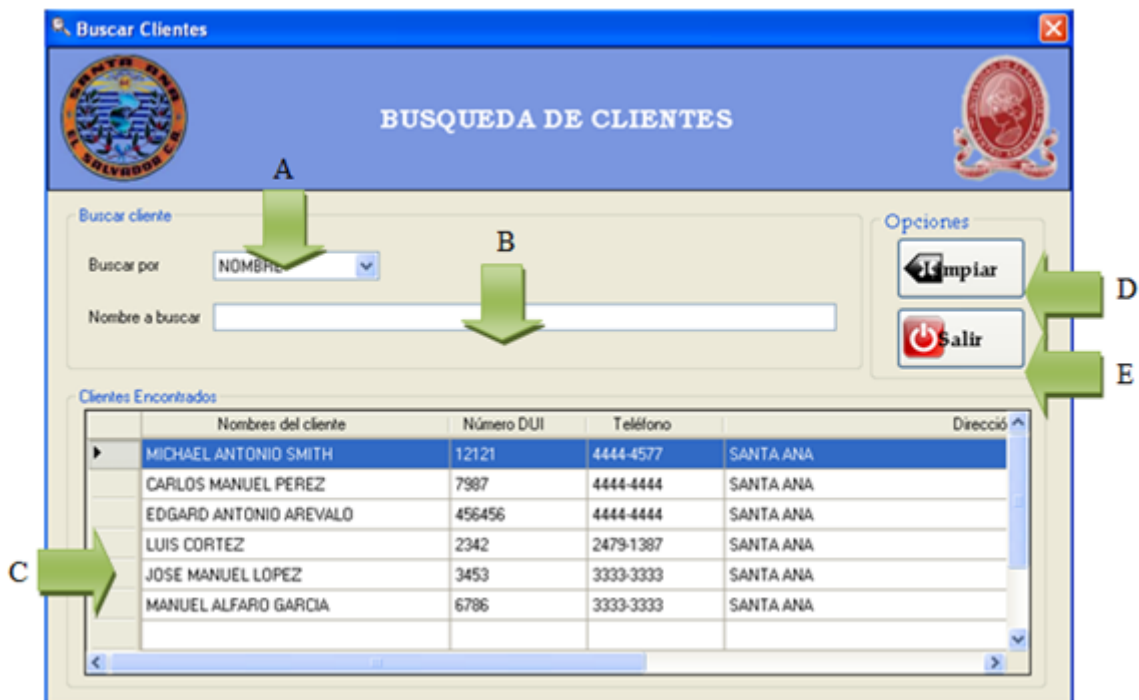


Figura 6.1.75

A continuación en la figura 6.1.54 se describe cada una de las partes de la pantalla de búsqueda de clientes:

- A. Especifica si la búsqueda se realizara mediante nombre o mediante documento único de identidad DUI
- B. En este lugar se debe escribir ya sea el nombre o el DUI según lo que haya seleccionado en el literal A.
- C. Muestra el listado de clientes ingresados al sistema
- D. Permite limpiar lo que se halla escrito en la caja de texto, para seleccionar un cliente e insertarlo en la factura basta con hacer clic sobre este y automáticamente se agregara al formulario de la ISAM.
- E. Permite salir de la sección de búsqueda de clientes.

AGREGAR SERVICIOS A LA FACTURA

En la figura 6.1.76 se muestra la forma de agregar servicios a la factura que se está emitiendo.



Figura 6.1.76

- A. En este espacio se debe escribir el nombre del impuesto el cual será autocompletado por el sistema.
- B. Se debe especificar la cantidad del mismo servicio que se agregara a la factura.
- C. Muestra todos los servicios ingresados al sistema que están disponibles para agregar a la factura. Para seleccionar un servicio basta simplemente hacer clic sobre el y este se cargara a la factura.
- D. Permite cancelar la operación actual retornando a la pantalla de la factura.
- E. Permite salir de la aplicación.

AGREGAR IMPUESTOS A LA FACTURA

En la figura 6.1.77 se muestra la forma de agregar impuestos a la factura que se está emitiendo.

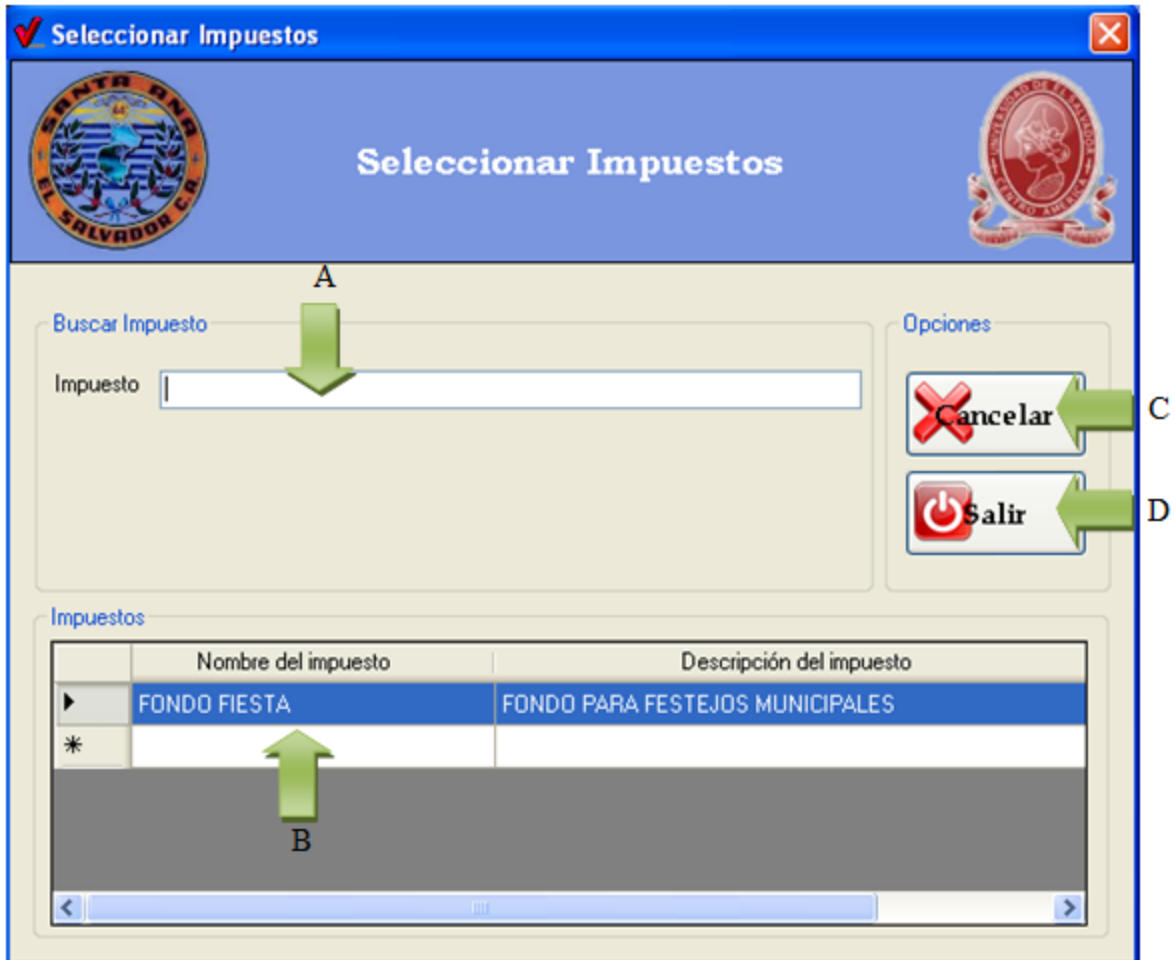


Figura 6.1.77

- Se debe escribir el nombre del impuesto que se desea agregar, dicho nombre a medida que se escriba se auto complementara automáticamente.
- Se muestran los impuestos agregados al sistema, para agregar un impuesto a la factura basta simplemente con hacer clic sobre este.
- Cancela la operación y cierra el formulario.
- Permite cerrar la ventana.

PROCESO DE ANULACION DE UNA ISAM

En la figura 6.1.78 se muestra la forma de acceder a la sección de anulación de facturas:

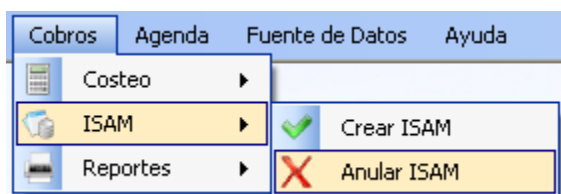


Figura 6.1.78

FORMA DE ANULAR UNA ISAM

En la figura 6.1.79 se muestra la pantalla para anular facturas, en dicha figura se explican cada uno de los campos para dicho procesos.

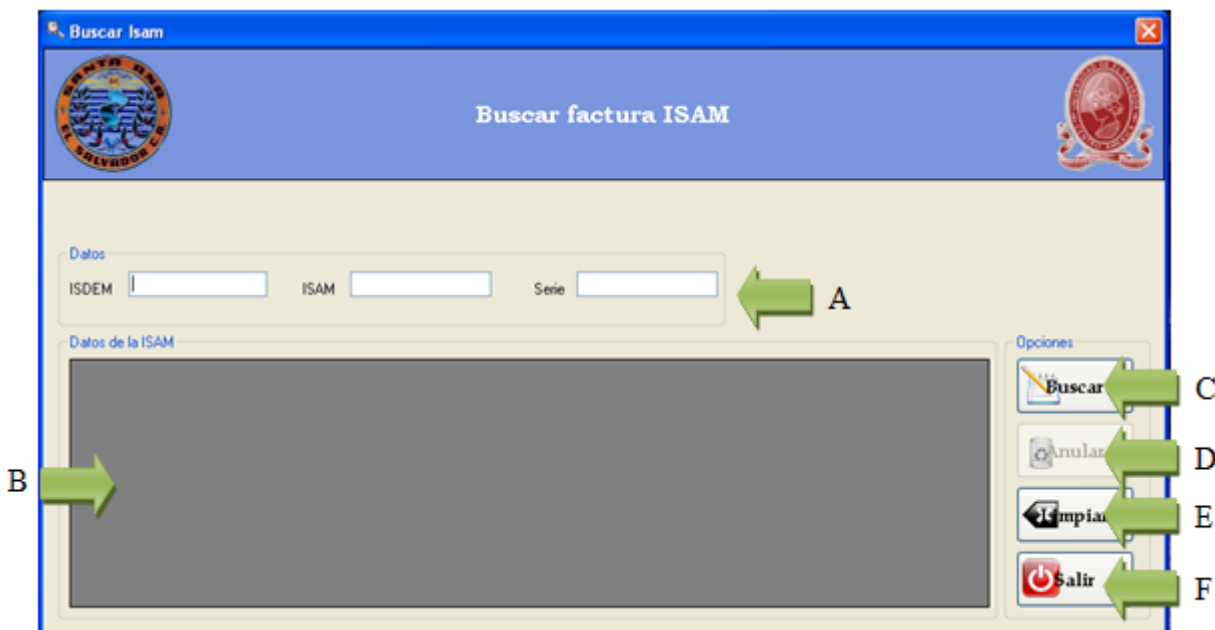


Figura 6.1.79

- A. En esta sección de la pantalla se debe introducir el numero de ISDEM, ISAM y la serie respectivamente los cuales corresponden a la ISAM que se desea anular.
- B. Aparecen los datos de la factura que se eliminara, en caso que la factura no exista o que ya fue anulada se mostrara un mensaje avisándole sobre evento.
- C. Permite buscar la factura con los datos que se ingresaron en el lit. A
- D. Una vez encontrada la factura se activara el botón anular el cual cargara otra ventana para realizar la anulación.
- E. Permite limpiar las cajas de texto e ingresar nuevos valores.

F. Permite cerrar la ventana.

En la figura 6.1.80 se muestra una factura que será eliminada pudiendo observar el botón Anular activo.

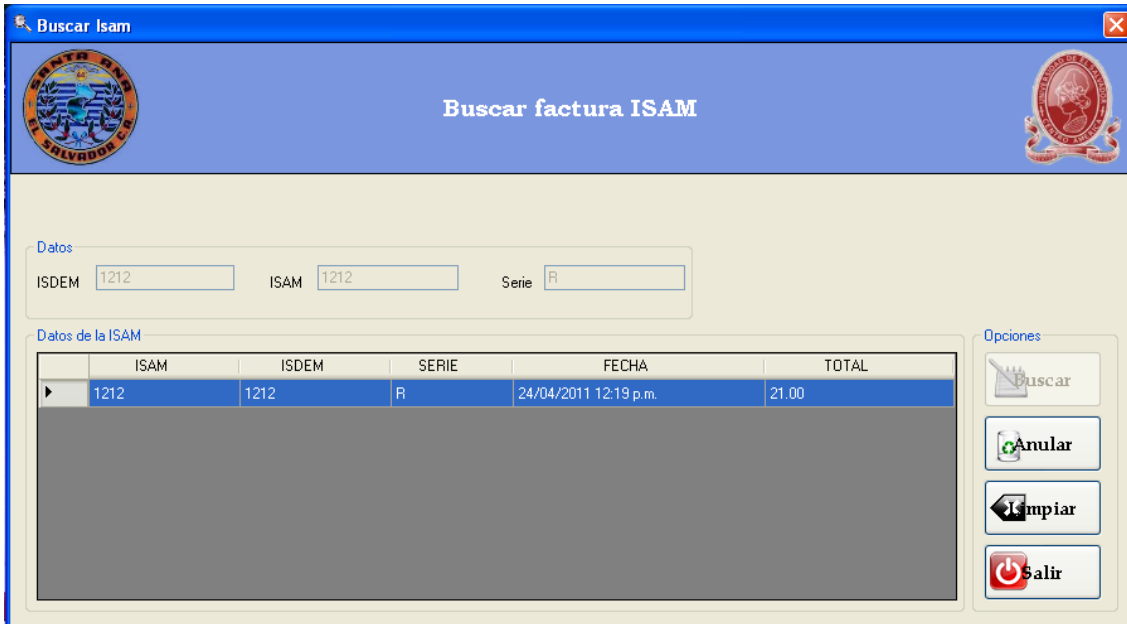


Figura 6.1.80

En la figura 6.1.81 se muestra la pantalla que emerge luego de hacer clic sobre el botón Anular, en dicha pantalla únicamente basta con especificar el motivo por el cual será anulada la ISAM y hacer clic en el botón Aceptar. Luego de haber hecho clic sobre el botón Aceptar emergerá un mensaje indicando que el proceso de anulación se realizó de forma exitosa, dicho mensaje se muestra en la figura 6.1.82

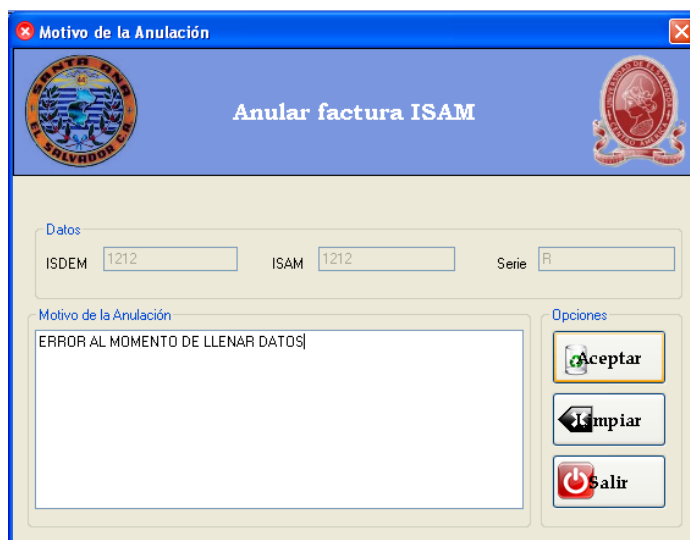


Figura 6.1.81

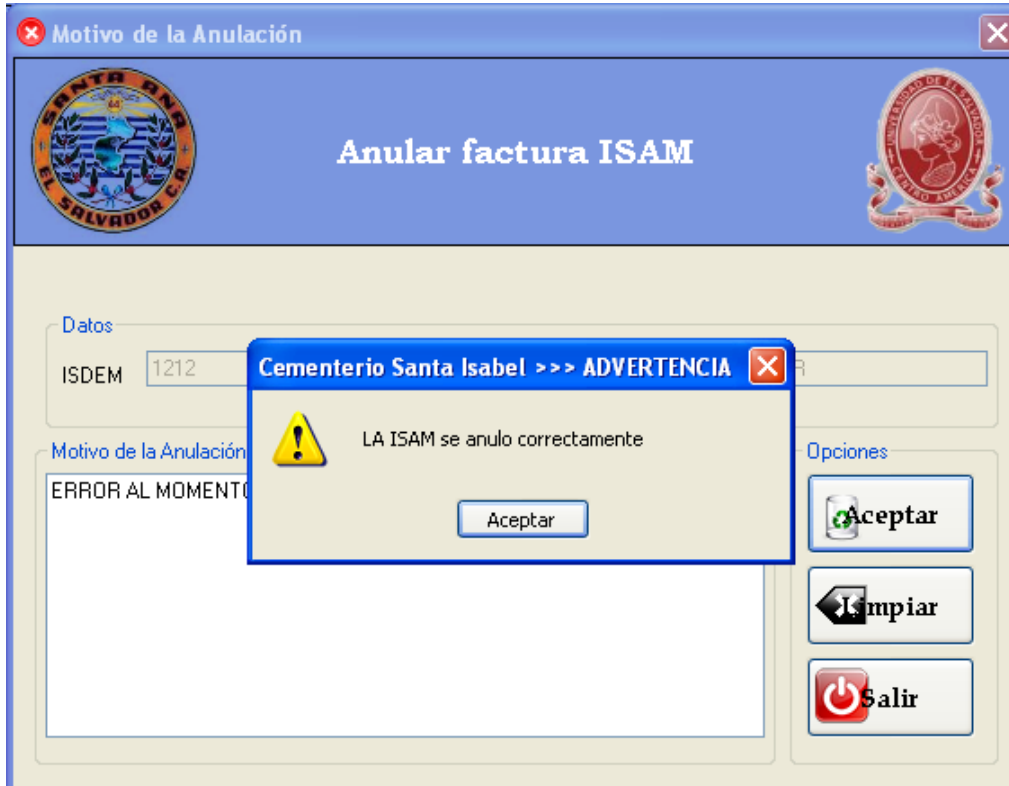


Figura 6.1.82

INFORMES DE ISAM EMITIDAS Y ANULADAS

En la figura 6.1.83 se muestra el menú para generar informes.

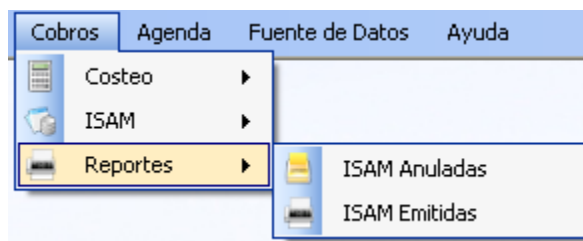


Figura 6.1.83

Dicho menú contiene 2 reportes, el primero de ellos permite visualizar todas las ISAM anuladas en base a un periodo de tiempo, mostrando los datos de la misma como la fecha y hora en la que esta fue agregada.

En el caso de la segunda opción permite visualizar todas las facturas ISAM emitidas filtradas por un periodo de tiempo. Cabe mencionar que al igual que el reporte anterior

este también muestra los datos de la ISAM así como la fecha y hora en que esta fue creada.

En la figura 6.1.84 se muestra el reporte que devuelve las ISAM anuladas.

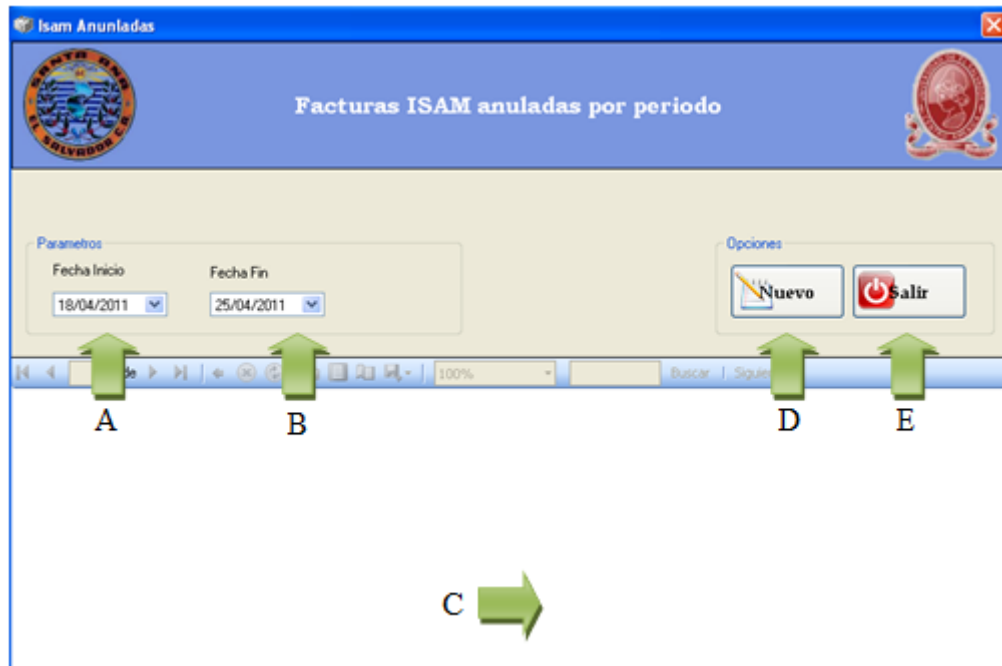


Figura 6.1.84

- A. Permite seleccionar la fecha a partir de la cual se desean ver las facturas.
- B. Permite seleccionar la fecha hasta la cual se desean ver las facturas.
- C. Muestra la información de todas las facturas anuladas en base a las fechas indicadas anteriormente.
- D. Genera el reporte.
- E. Permite cerrar el reporte y regresar al menú anterior.

En la figura 6.1.85 se muestra el reporte de todas las ISAM emitidas en base a un periodo, cabe mencionar que dicho reporte es similar al de las facturas anuladas.

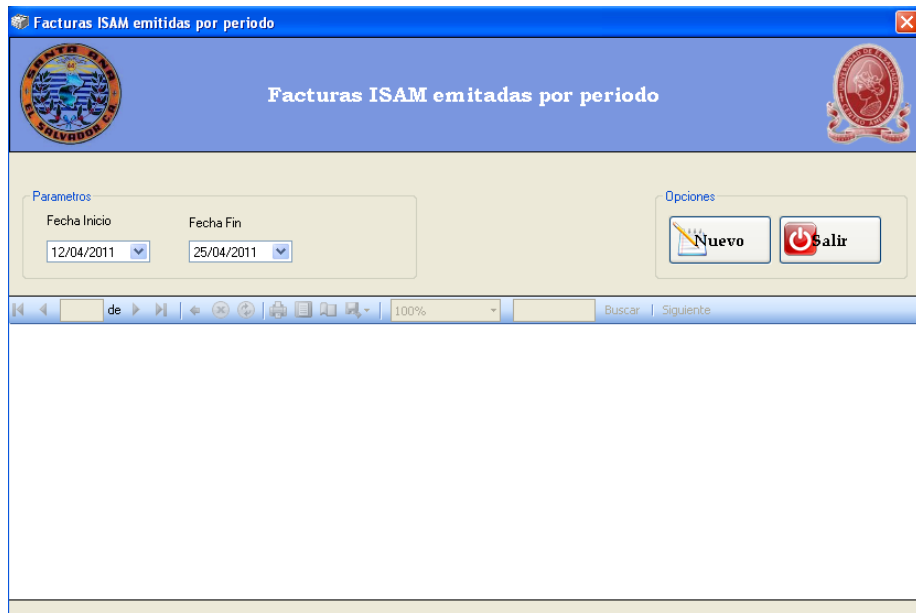


Figura 6.1.85

MENÚ AGENDA

Permite administrar todo lo relacionado a los planes de trabajo y las actividades por las cuales están conformados estos planes, incluyendo una agenda con la calendarización de las actividades.

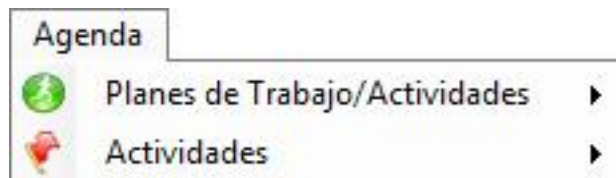


Figura 6.1.86 Agenda

OPCIÓN PLANES DE TRABAJO/ACTIVIDADES

Permite administrar lo referente a la creación y modificación de planes de trabajo y actividades, así como también permite agregar empleados y actividades al plan de trabajo.



Figura 6.1.87 Sub-opciones de trabajo/actividades

Sub-opción Crear/Modificar

Permite la creación y modificación de los planes de trabajo, así como también la creación y modificación de las actividades del cementerio.

Pestaña de Planes de Trabajo

Permite administrar los planes de trabajo del cementerio, es decir crearlos y modificarlos.



Figura 6.1.88 Creación y modificación de planes de trabajo

A. Planes de Trabajo indica que la pestaña activa corresponde a la administración de los planes de trabajo.

B. Los datos del plan de trabajo necesarios para poder registrar a un usuario dentro del sistema de información o modificar uno se ingresan acá. Para poder registrar correctamente el plan deberá llenar todos los campos solicitados correctamente, siguiendo las siguientes directrices:

- 1) El nombre del plan de trabajo y la descripción no pueden contener caracteres especiales.
- 2) El año del plan de trabajo solamente puede ser en el rango del año actual o el siguiente año, pero no puede pasar de estos límites.

Si se diera el caso que el nombre de plan que estamos ingresando ya existe, el sistema nos mostrará el siguiente mensaje de error:

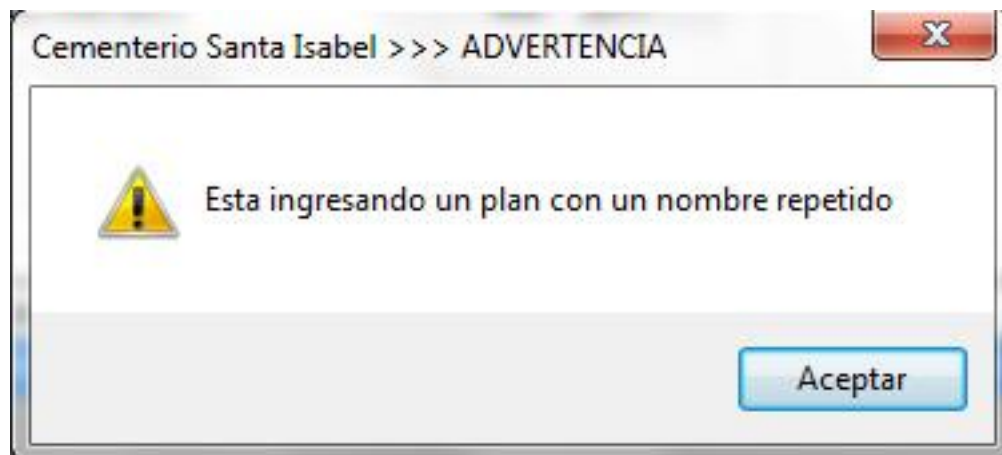


Figura 6.1.89 Mensaje de Error de nombre del plan de trabajo

Otro caso es aquel en el cual el nombre de plan que estamos modificando ya lo tiene otro plan, en cuyo caso el sistema nos mostrara el siguiente mensaje:

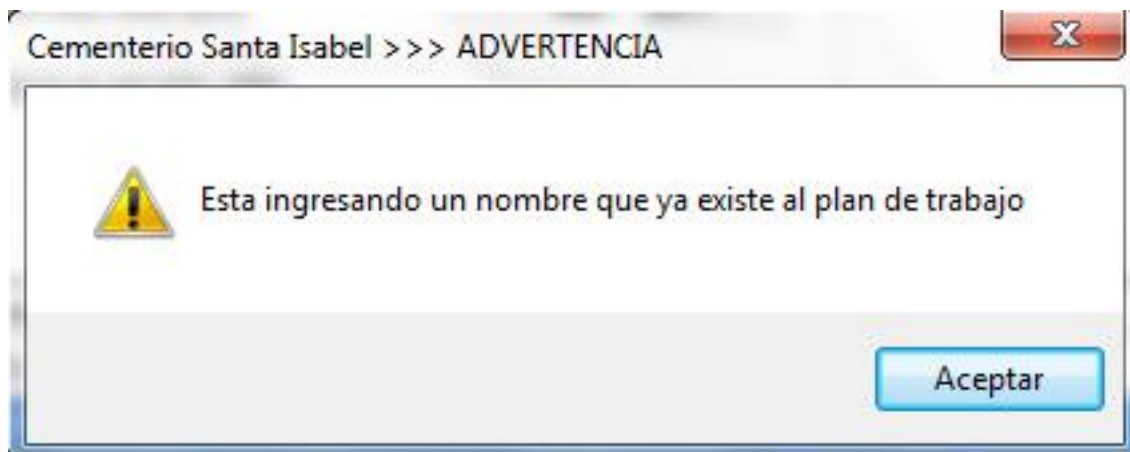


Figura 6.1.90 Mensaje de Error de nombre del plan de trabajo

En caso que el año de plan que estamos ingresando ya existe, el sistema nos mostrará el siguiente mensaje de error:

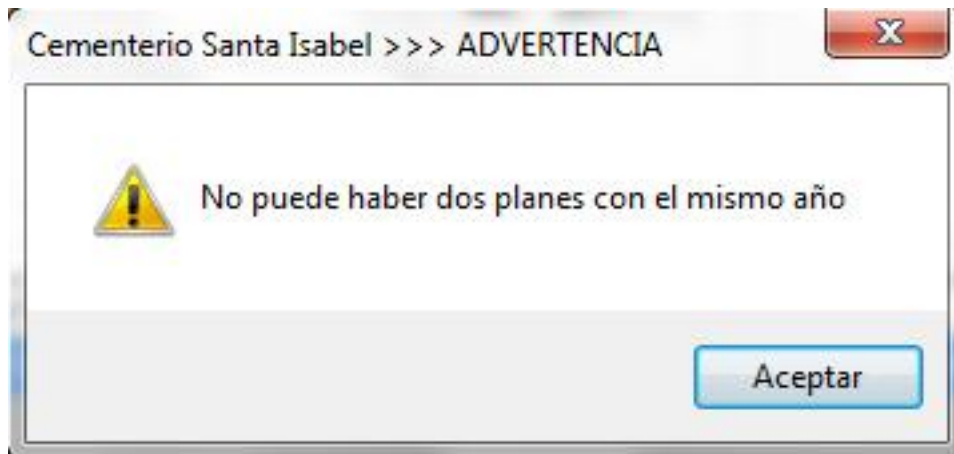


Figura 6.1.91 Mensaje de Error de año repetido

En caso que se omita alguna de las directrices anteriores, obtendríamos el siguiente mensaje de error al presionar el botón guardar:

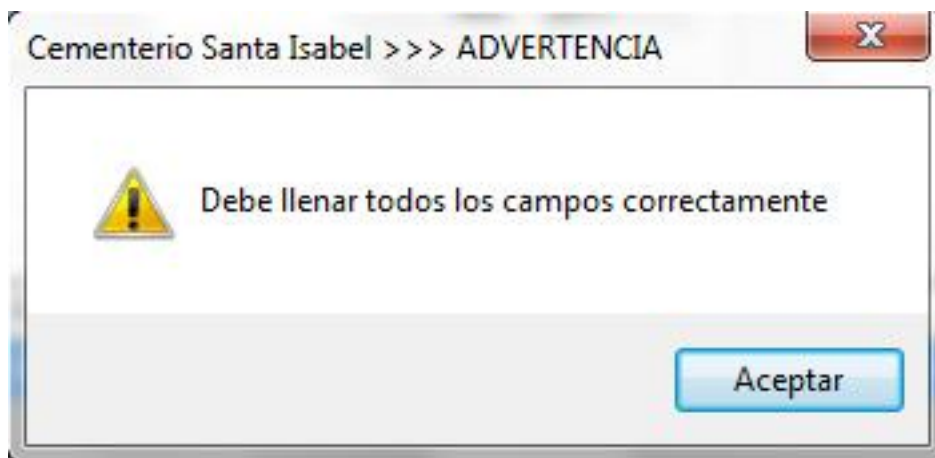


Figura 6.1.92 Mensaje de Error de Plan de Trabajo

C. Acá se muestran los planes de trabajo registrados en el sistema; si se desea modificar algún plan de trabajo, debe seleccionarse de esta lista, pero solamente podrán ser modificados aquellos planes de trabajo del año actual o posterior.

D. Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.88 con el botón habilitado y los demás campos se encontrarán inhabilitados, por lo que al hacer clic en este botón se habilitarán el listado de los planes de trabajo así como los campos para ingresar o modificar planes de trabajo.

E. Representa al botón guardar (se puede hacer clic en este botón con la tecla Enter), que deberá presionarse cuando hayamos proporcionado todos los datos necesarios para guardar un nuevo plan de trabajo o para modificar uno existente. En caso que hayamos seguido las directrices antes mencionadas, en el caso de un nuevo plan, se mostrará en

pantalla un mensaje preguntándonos si queremos guardar el plan (estos no se pueden eliminar) tal como se muestra en la siguiente imagen:

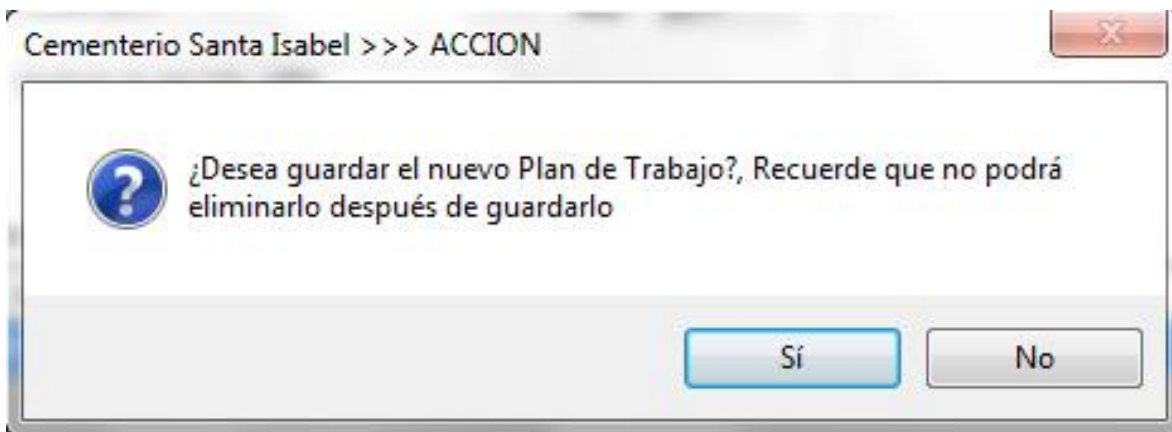


Figura 6.1.93 Mensaje de preguntando si queremos guardar el plan

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que el plan se guardó correctamente con el siguiente mensaje:

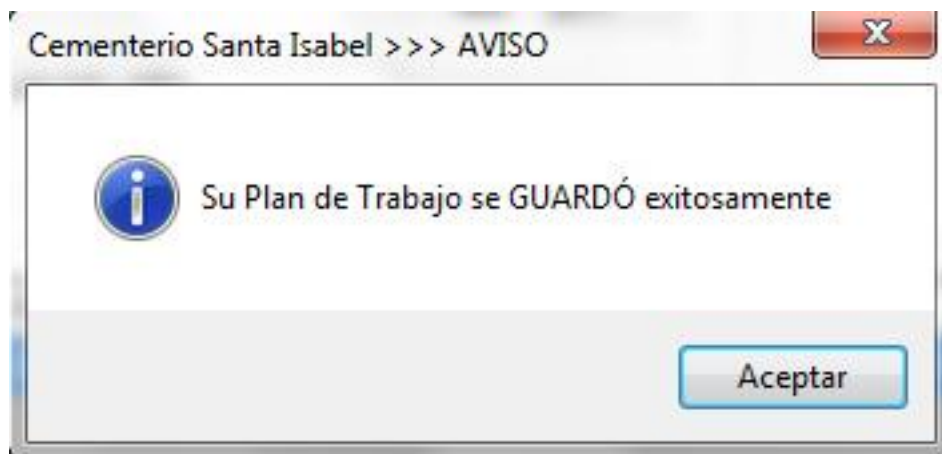


Figura 6.1.94 Mensaje confirmando al plan guardado

Si el caso es la modificación de los datos de un plan, se mostrara la siguiente pregunta en forma de mensaje:

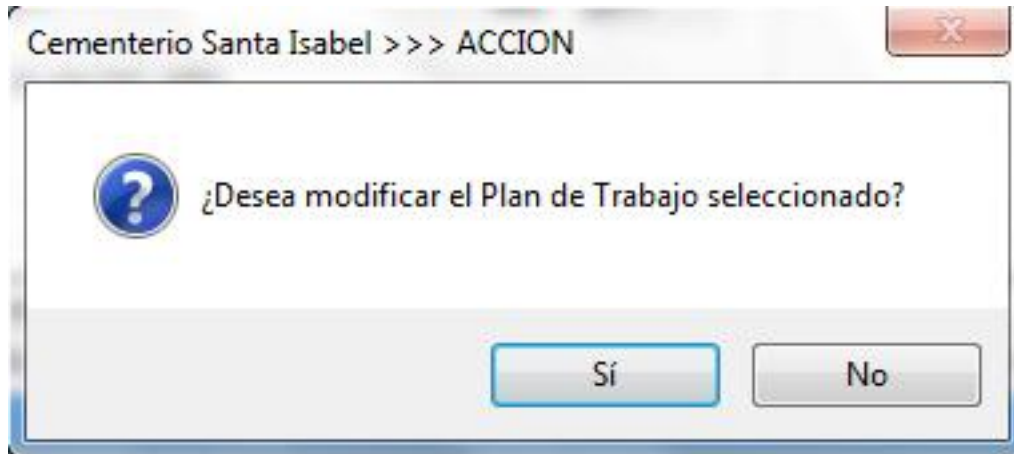


Figura 6.1.95 Mensaje preguntando si queremos modificar el plan

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que el plan se modificó correctamente con el siguiente mensaje:

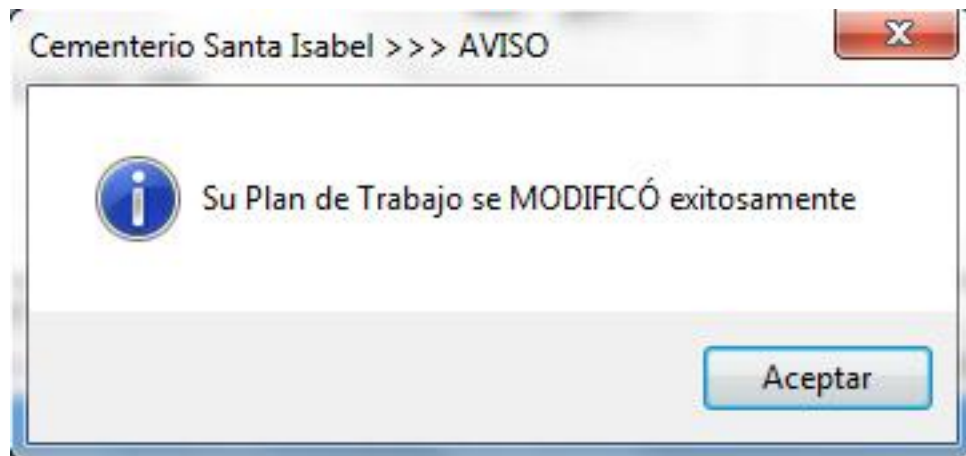


Figura 6.1.96 Mensaje confirmando al plan modificado

F. Botón que permite guardar las modificaciones de un registro y que por defecto se encuentra inhabilitado, ya que para que este se encuentre activo primero deberá seleccionar algún plan del listado y si este plan corresponde al año actual o posterior, se habilitara este botón, luego si usted hace clic en este botón se habilitara la modificación de los datos del plan (todos menos el año), a la vez que se habilitara el botón guardar para poder guardar los cambios.

G. Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el siguiente mensaje:

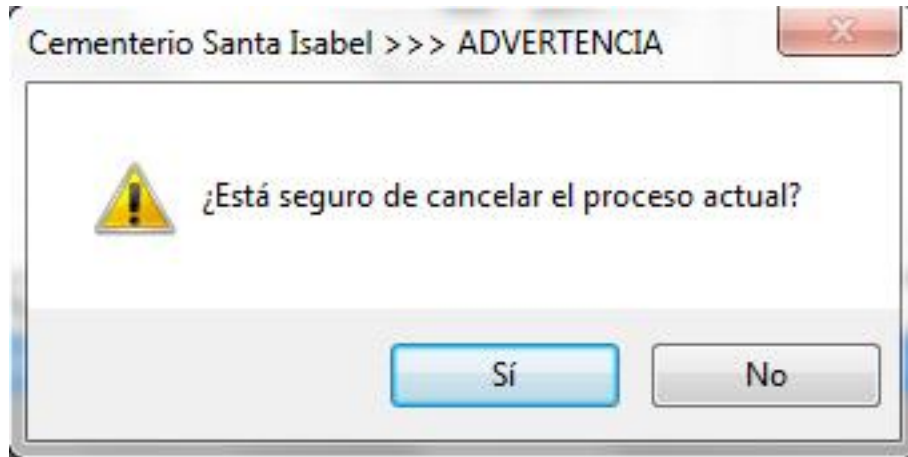


Figura 6.1.97 Mensaje preguntando si queremos cancelar la acción

H. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.88, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán. En caso que queramos salir nos mostrara el siguiente mensaje:

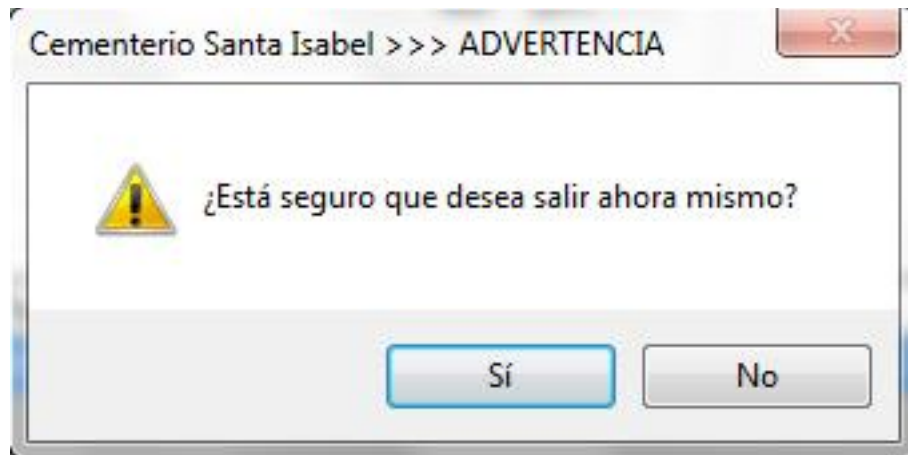


Figura 6.1.98 Mensaje preguntando si queremos salir del formulario

Pestaña de Actividades

Permite administrar las actividades del cementerio, es decir crearlas y modificarlas.

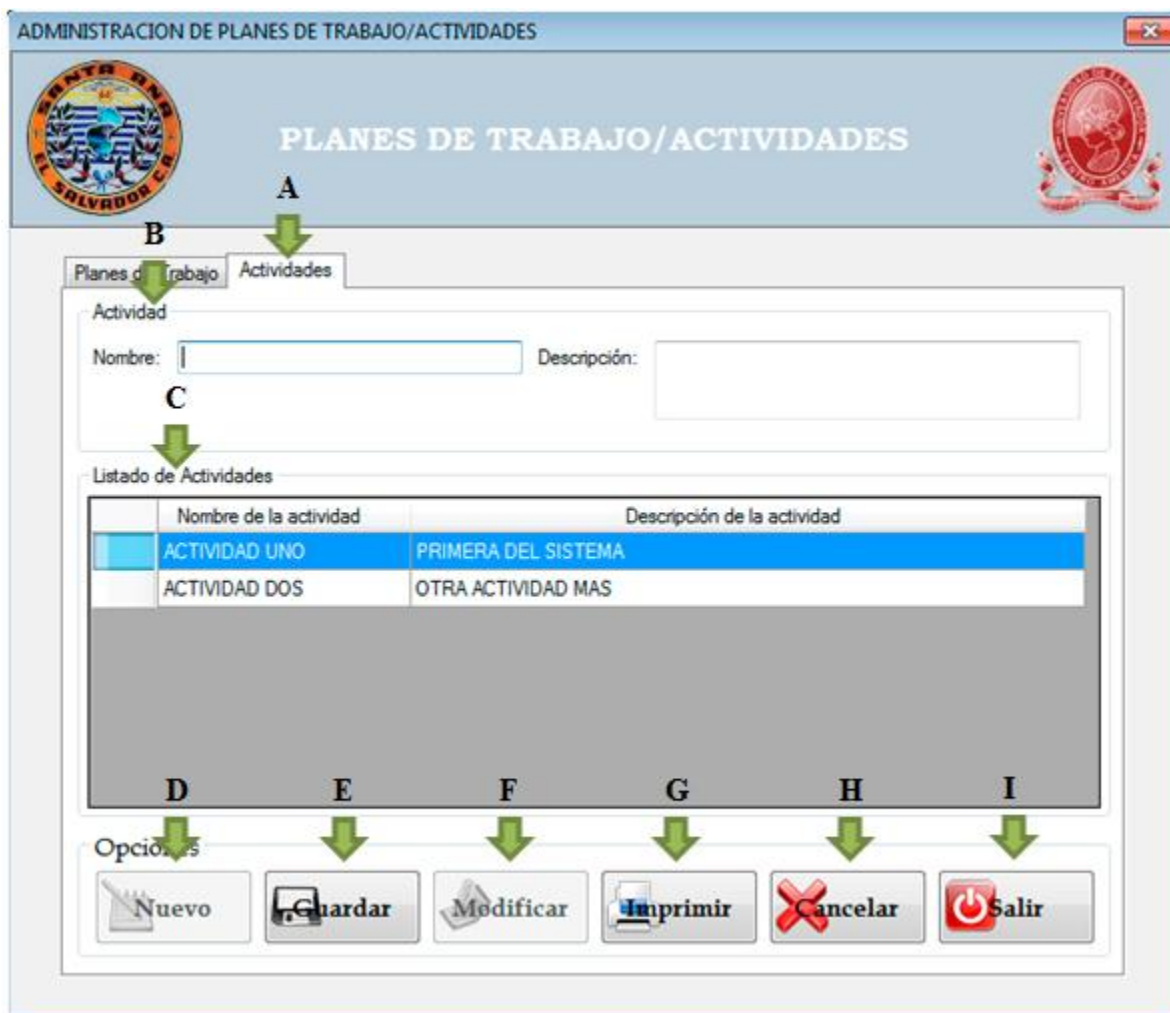


Figura 6.1.99 Creación y modificación de actividades

A. Actividades indica que la pestaña activa corresponde a la administración de las actividades.

B. Los datos de la actividad necesarios para poder registrar a una actividad dentro del sistema de información o modificar una se ingresan acá. Para poder registrar correctamente la actividad deberá llenar todos los campos solicitados correctamente, siguiendo las siguientes directrices:

- 1) El nombre de la actividad y descripción no puede contener caracteres especiales.

Si se diera el caso que el nombre de la actividad que estamos ingresando ya existe, el sistema nos mostrará el siguiente mensaje de error:

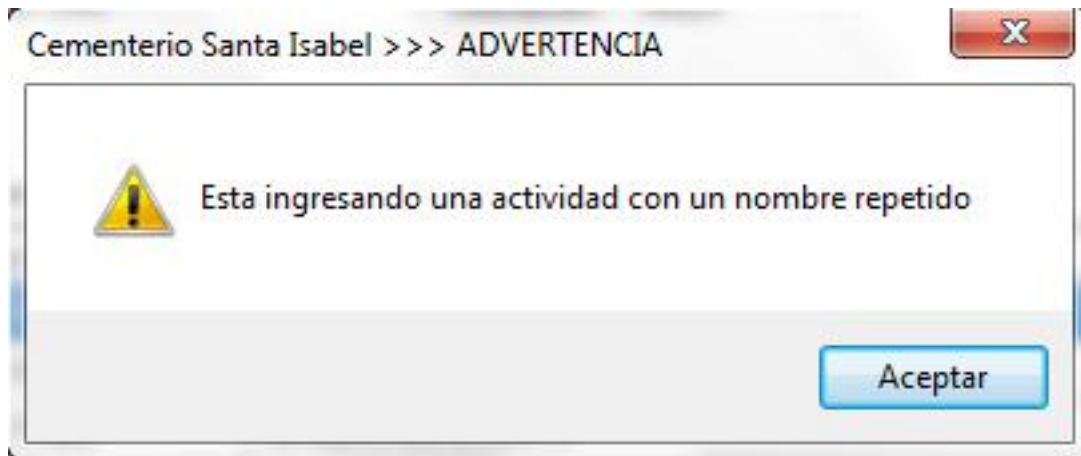


Figura 6.1.100 Mensaje de Error de nombre de la actividad

Otro caso es aquel en el cual el nombre de la actividad que estamos modificando ya lo tiene otra actividad, en cuyo caso el sistema nos mostrara el siguiente mensaje:

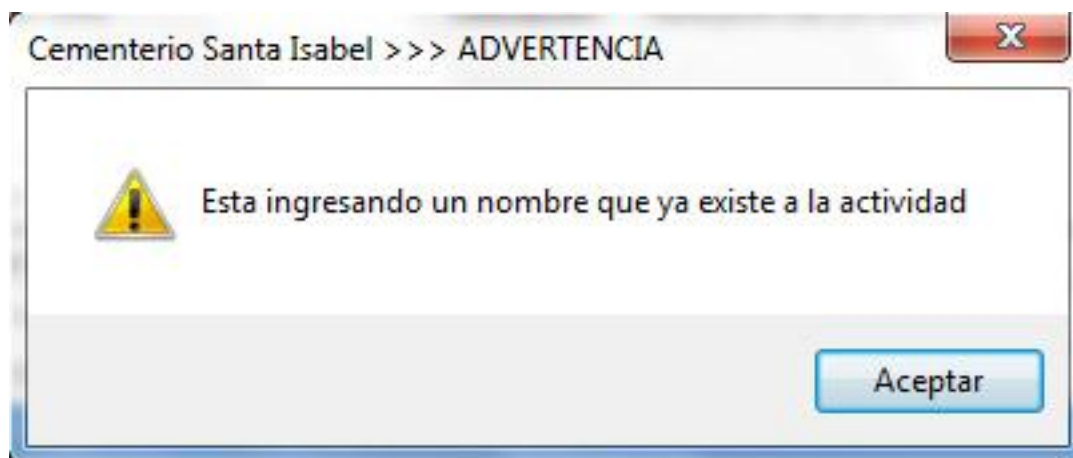


Figura 6.1.101 Mensaje de Error de nombre de la actividad

En caso que se omita alguna de las directrices anteriores, obtendríamos el mensaje de error correspondiente a la figura 6.1.92

C. Aquí se muestran las actividades registradas en el sistema; si se desea modificar alguna actividad, debe seleccionarse de esta lista.

D. Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.99 con el botón habilitado y los demás campos se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran el listado de las actividades así como los campos para ingresar o modificar actividades.

E. Representa al botón guardar (se puede hacer clic en este botón con la tecla Enter), que deberá presionarse cuando hayamos proporcionado todos los datos necesarios para guardar una nueva actividad o para modificar una existente. En caso que hayamos

seguido las directrices antes mencionadas, en el caso de una nueva actividad, se mostrará en pantalla un mensaje preguntándonos si queremos guardar la actividad tal como se muestra en la siguiente imagen:

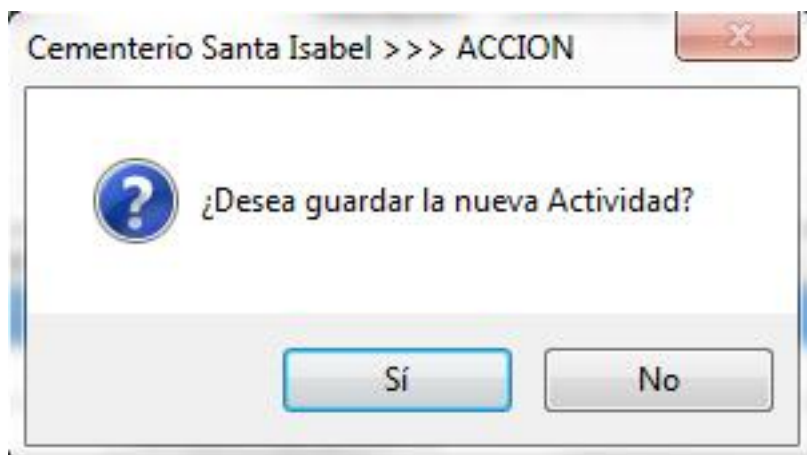


Figura 6.1.102 Mensaje preguntando si queremos guardar la actividad

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que la actividad se guardó correctamente con el siguiente mensaje:

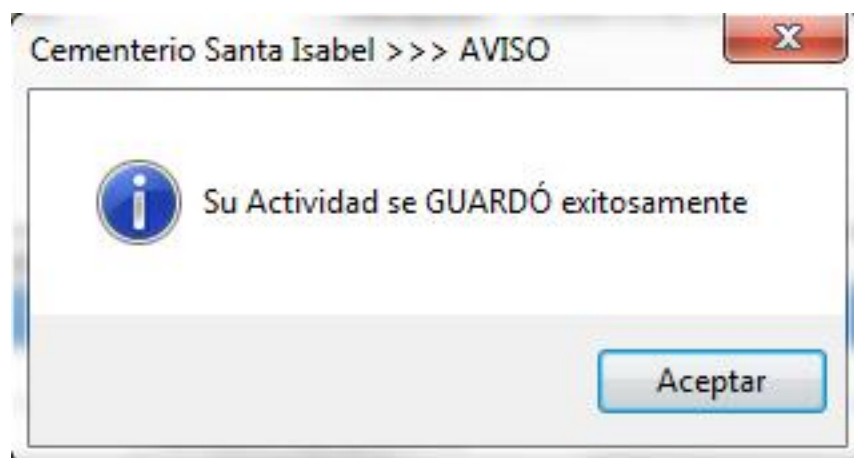


Figura 6.1.103 Mensaje confirmando la actividad guardada

Si el caso es la modificación de los datos de una actividad, se mostrara la siguiente pregunta en forma de mensaje:

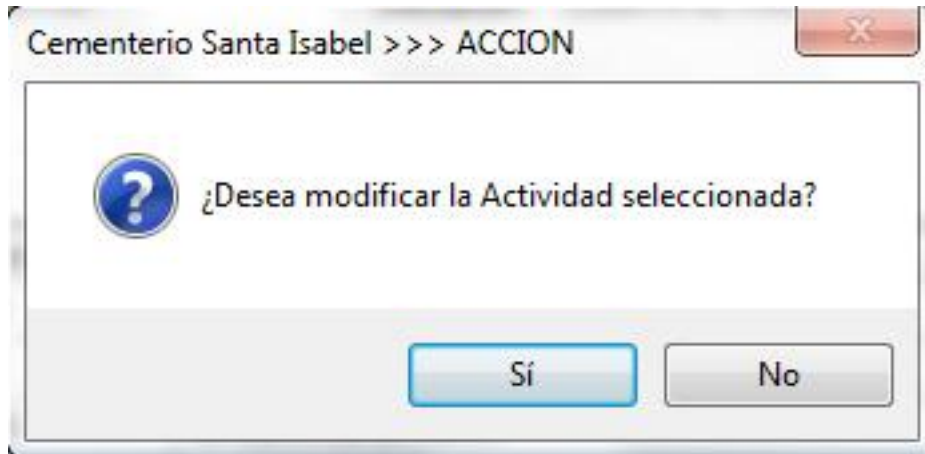


Figura 6.1.104 Mensaje preguntando si queremos modificar la actividad

Si respondemos afirmativamente a la pregunta anterior, nos confirmara que la actividad se modificó correctamente con el siguiente mensaje:

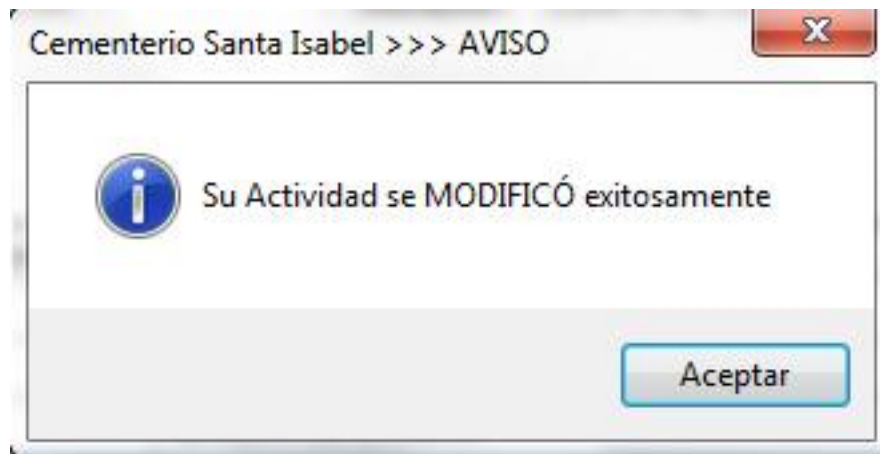


Figura 6.1.105 Mensaje confirmando la actividad modificada

F. Botón que permite guardar las modificaciones de un registro y que por defecto se encuentra inhabilitado, ya que para que este se encuentre activo primero deberá seleccionar alguna actividad del listado, luego se habilitara este botón, a continuación si usted hace clic en este botón se habilitara la modificación de los datos de la actividad, a la vez que se habilitara el botón guardar para poder guardar los cambios.

G. Botón que permite generar un reporte con las actividades del cementerio (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:

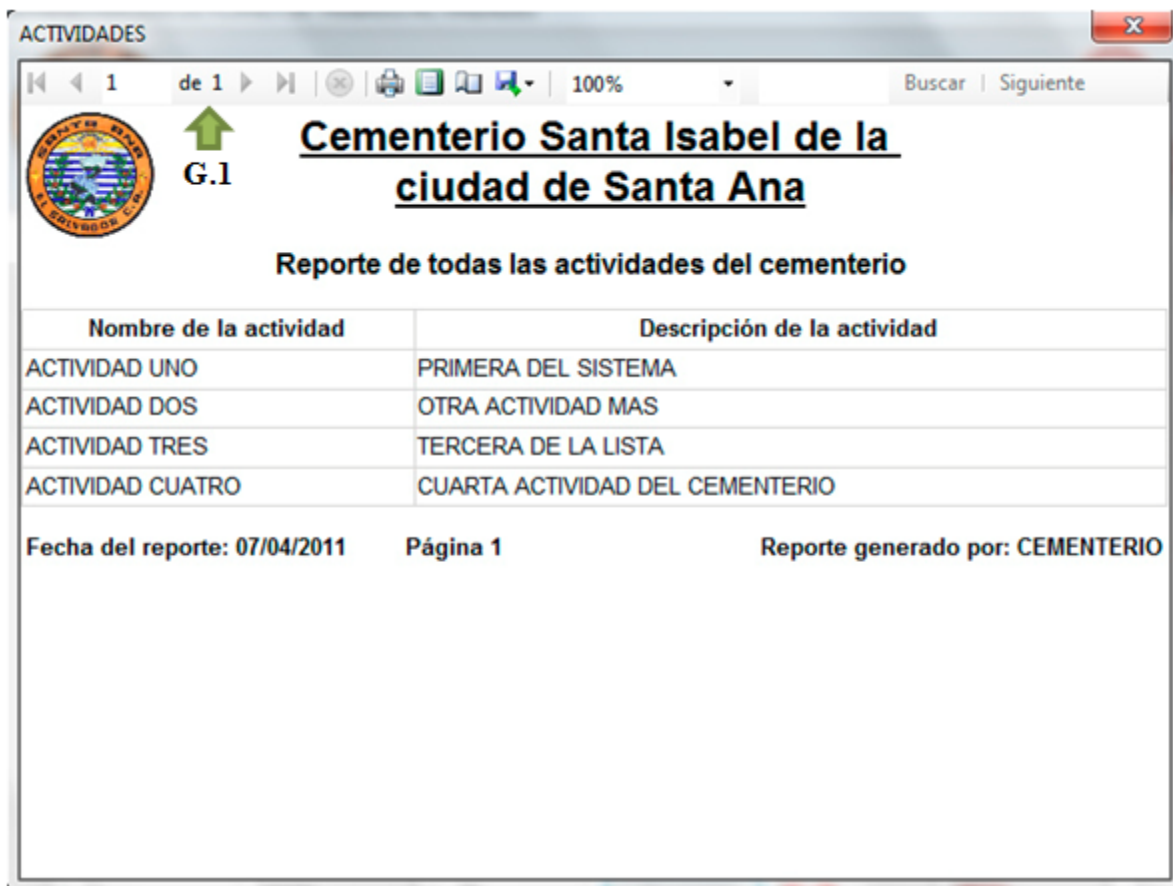


Figura 6.1.106 Reporte de las actividades del cementerio

G.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

H. Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el mensaje de la figura 6.1.97

I. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.99, sin embargo debemos aclarar que si no hemos

guardado un registro y presionamos este botón saldremos de la pantalla y los datos digitados se perderán. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Sub-opción Agregar Empleados/Actividades

Permite agregar empleados al plan de trabajo, así como también agregar actividades al plan de trabajo.

Pestaña de Empleados del Plan de Trabajo

Permite agregar empleados a los planes de trabajo.

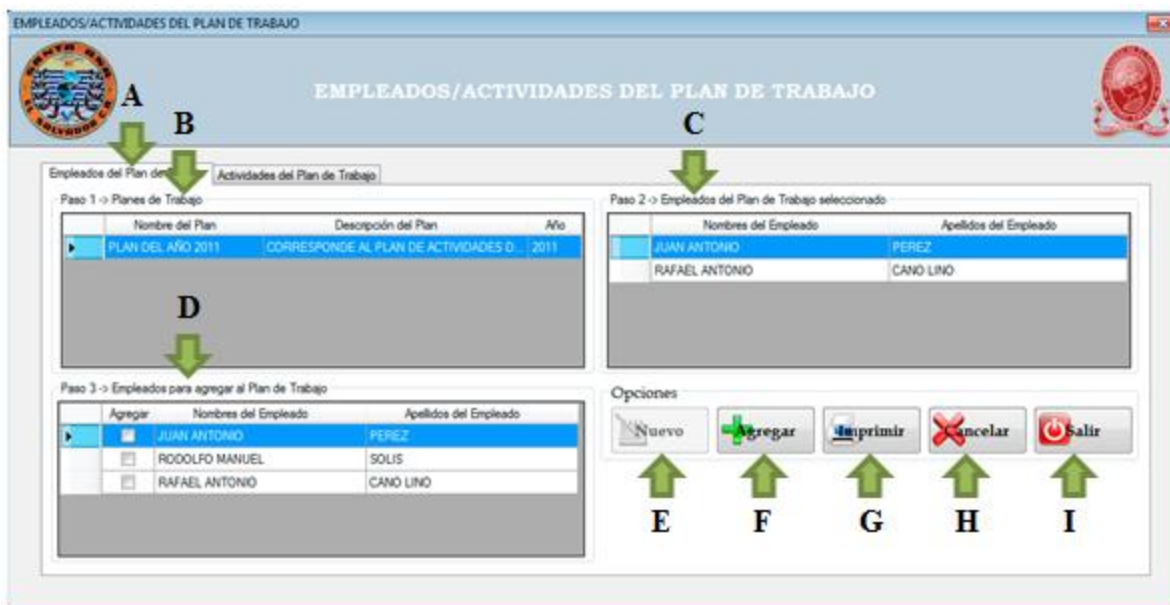


Figura 6.1.107 Agregar empleados al plan de trabajo

- A.** Esta pestaña activa corresponde a la agregación de empleados al plan de trabajo.
- B.** Muestra el listado de los planes de trabajo disponibles, debe de seleccionar un plan de trabajo para poder agregarle empleados, pero solamente podrá agregarle empleados a los planes de trabajo cuyo año corresponda ya sea al año actual o al posterior a este.
- C.** Muestra el listado de los empleados que están dentro del plan de trabajo que se ha seleccionado, este listado sirve solamente para visualizar a estos empleados, no tiene otra funcionalidad más.
- D.** Muestra el listado de los empleados activos del cementerio, notara que al inicio de cada empleado hay una cajita la cual puede ser marcada, el objetivo acá es marcar la cajita correspondiente a los empleados que se desean agregar al plan de trabajo, pero cabe mencionar que solamente podrá marcar aquellos empleados que no están dentro del

plan de trabajo seleccionado, ya que el sistema no permite marcar a los empleados que ya están dentro del plan de trabajo, todo esto evita que se agreguen empleados duplicados evitando así posibles errores.

E. Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.107 con el botón habilitado y los demás campos se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran el listado los de planes de trabajo.

F. Representa al botón agregar (se puede hacer clic en este botón con la tecla Enter), el cual permite agregar empleados al plan de trabajo, este botón solamente estará habilitado si existe algún plan de trabajo seleccionado, si este plan es del año actual o posterior. Si hacemos clic en este botón y no marcamos ningún empleado nos mostrara el siguiente mensaje de error:

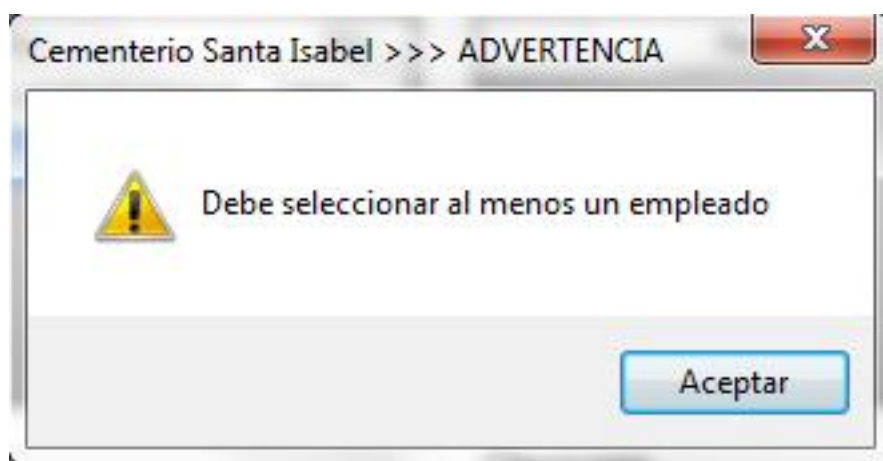


Figura 6.1.108 Mensaje de Error por no seleccionar empleados

Caso contrario, si marcamos al menos un empleado nos preguntara lo siguiente:

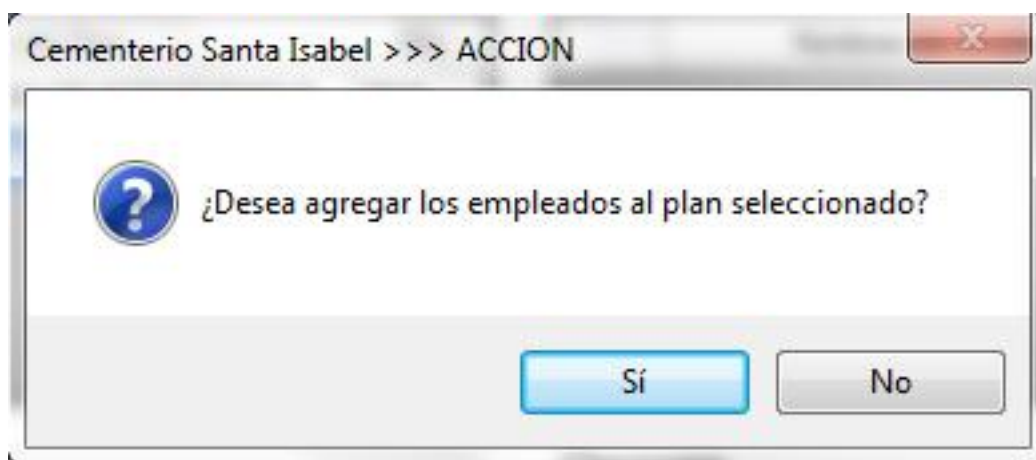


Figura 6.1.109 Mensaje preguntándonos por la agregación de empleados

Si respondemos afirmativamente la pregunta anterior, el sistema nos mostrara un mensaje confirmándonos que se han agregado los empleados al plan:

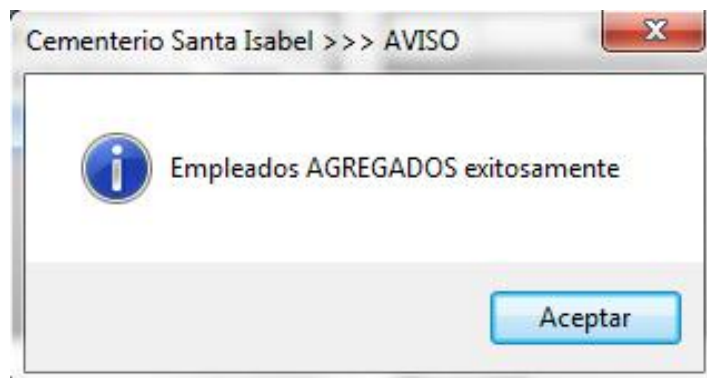


Figura 6.1.110 Mensaje de confirmación de empleados agregados

G. Botón que permite generar un reporte con los empleados del plan de trabajo (habilitado solo en el caso que existan empleados), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:

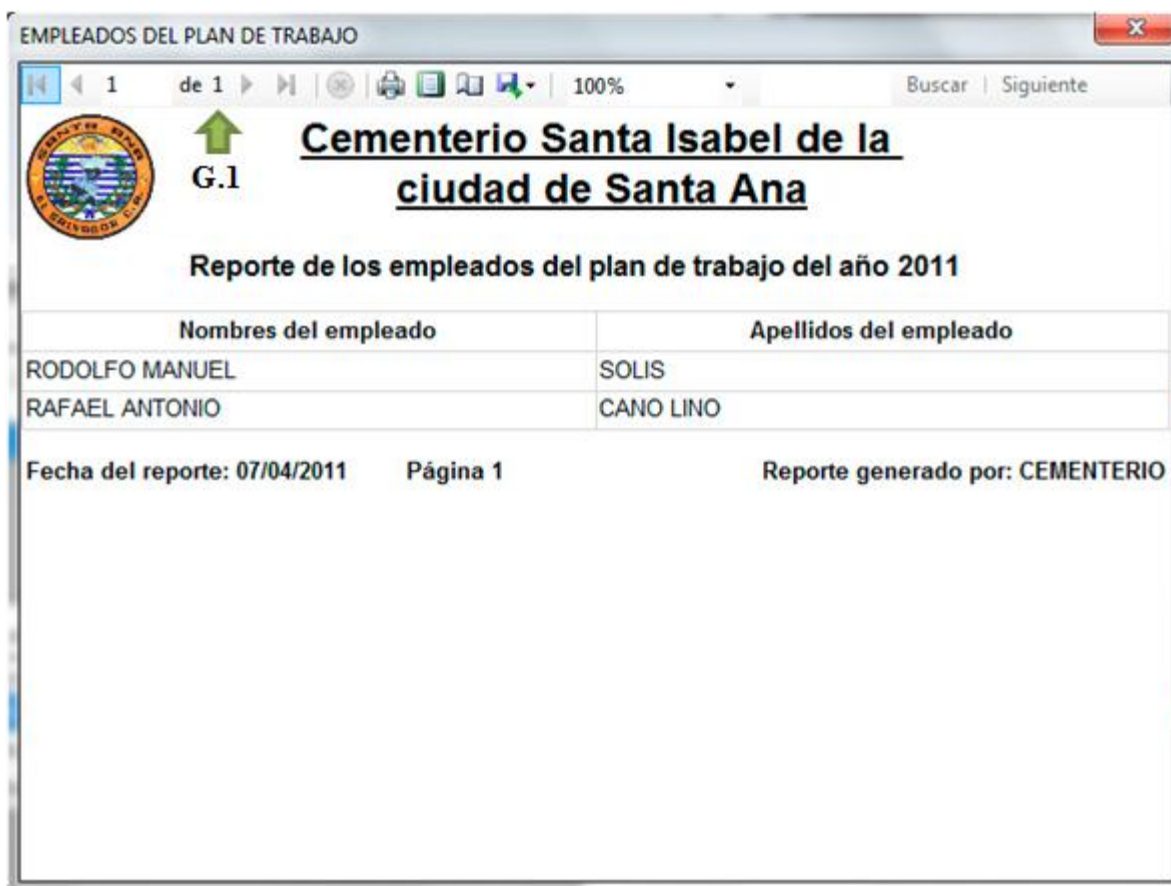


Figura 6.1.111 Reporte de los empleados del plan de trabajo

G.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

H. Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el mensaje de la figura 6.1.97

I. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.107, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos ingresados se perderán. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades del Plan de Trabajo

Permite agregar actividades a los planes de trabajo.



Figura 6.1.112 Agregar actividades al plan de trabajo

- A.** Esta pestaña activa corresponde a la agregación de actividades al plan de trabajo.
- B.** Muestra el listado de los planes de trabajo disponibles, debe de seleccionar un plan de trabajo para poder agregarle actividades, pero solamente podrá agregarle actividades a los planes de trabajo cuyo año corresponda ya sea al año actual o al posterior a este.
- C.** Muestra el listado de las actividades que están dentro del plan de trabajo que se ha seleccionado, este listado sirve solamente para visualizar a estas actividades, no tiene otra funcionalidad más.
- D.** Muestra el listado de las actividades del cementerio, notara que al inicio de cada actividad hay una cajita la cual puede ser marcada, el objetivo acá es marcar la cajita correspondiente a las actividades que se desean agregar al plan de trabajo, pero cabe mencionar que solamente podrá marcar aquellas actividades que no están dentro del plan de trabajo seleccionado, ya que el sistema no permite marcar a las actividades que ya están dentro del plan de trabajo, todo esto evita que se agreguen actividades duplicadas evitando así posibles errores.
- E.** Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.122 con el botón habilitado y los demás campos se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran el listado los de planes de trabajo.
- F.** Representa al botón agregar (se puede hacer clic en este botón con la tecla Enter), el cual agrega actividades al plan de trabajo, este botón solamente estará habilitado si existe algún plan de trabajo seleccionado, si este plan es del año actual o posterior. Si hacemos clic en este botón y no marcamos ninguna actividad nos mostrara el siguiente mensaje de error:

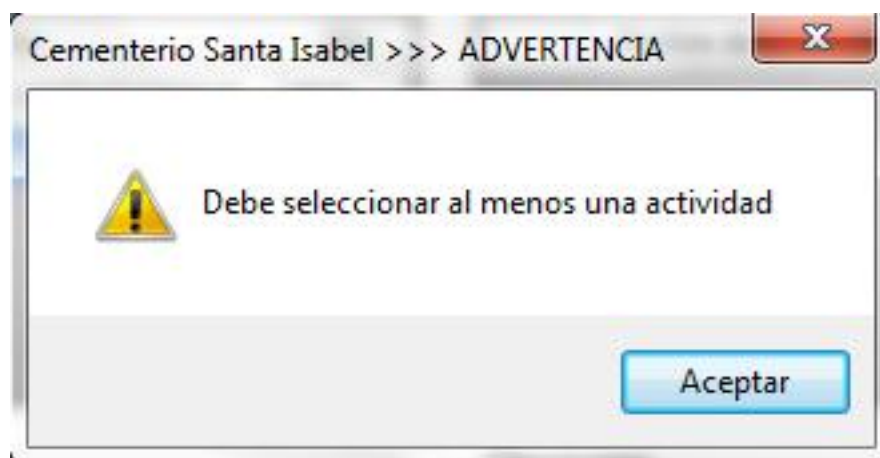


Figura 6.1.113 Mensaje de Error por no seleccionar actividades

Caso contrario, si marcamos al menos una actividad nos preguntara lo siguiente:

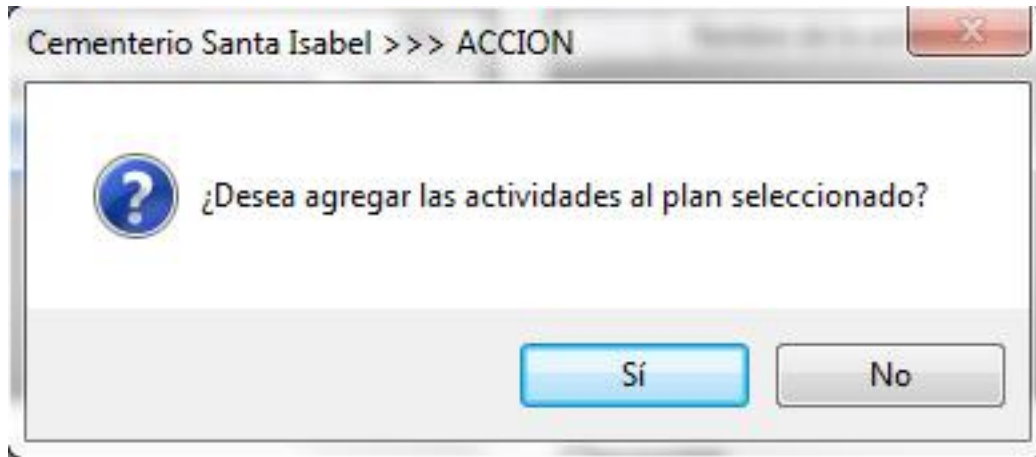


Figura 6.1.114 Mensaje preguntándonos por la agregación de actividades

Si respondemos afirmativamente la pregunta anterior, el sistema nos mostrara un mensaje confirmándonos que se han agregado las actividades al plan:

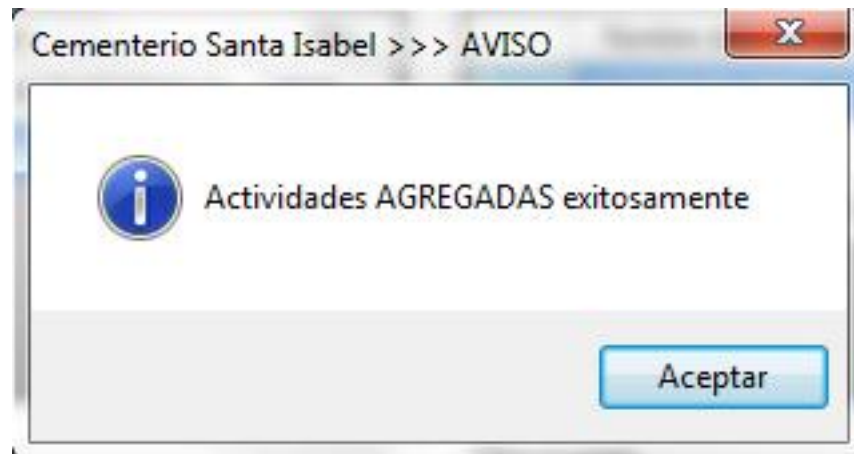


Figura 6.1.115 Mensaje de confirmación de actividades agregadas

G. Botón que permite generar un reporte con las actividades del plan de trabajo (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:

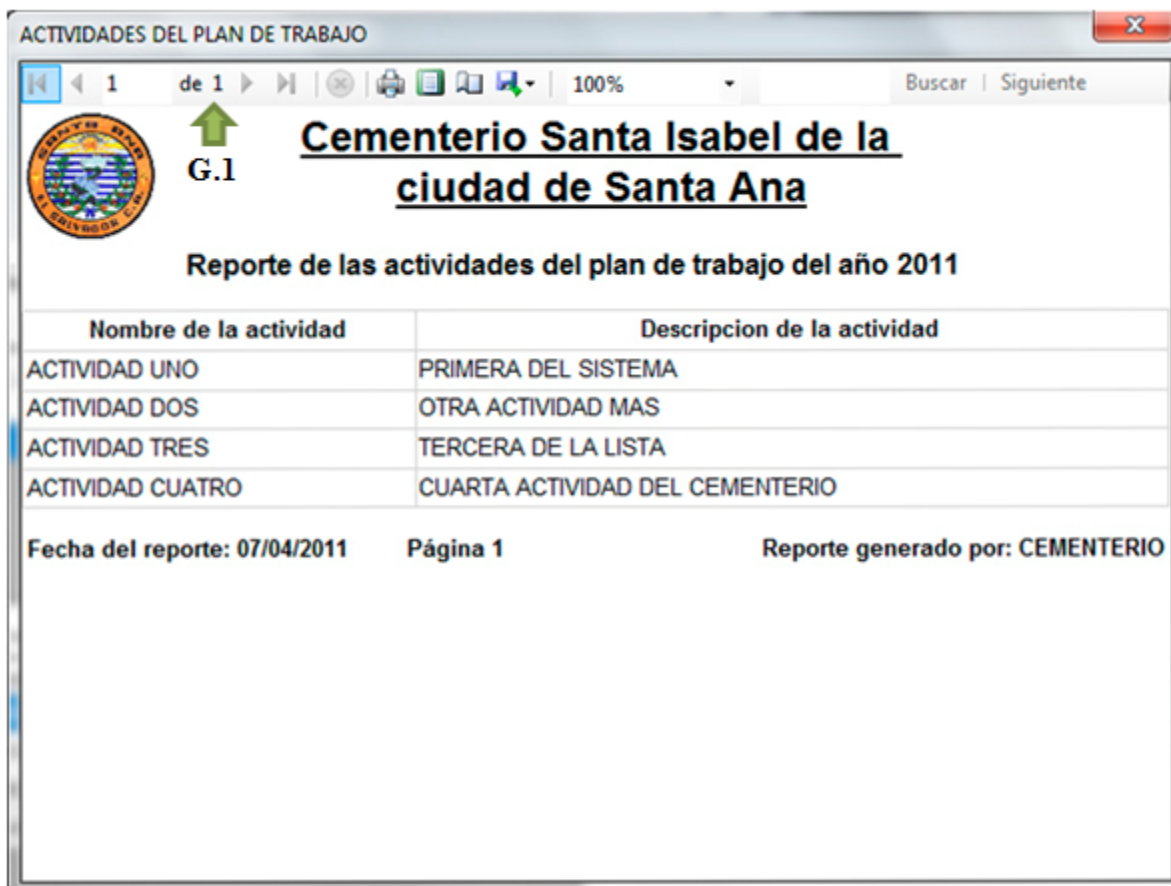


Figura 6.1.116 Reporte de las actividades del plan de trabajo

G.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- A) Navegar entre las diferentes páginas del reporte.
- B) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- C) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- D) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- E) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- F) Búsqueda dentro del reporte.

H. Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el mensaje de la figura 6.1.97

I. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.112, sin embargo debemos aclarar que si no hemos

guardado un registro y presionamos este botón saldremos de la pantalla y los datos ingresados se perderán. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

OPCIÓN ACTIVIDADES

Permite administrar todo lo referente a las actividades en general, pero también tiene que ver con los empleados en el papel de eliminación, así como de asignación de actividades.

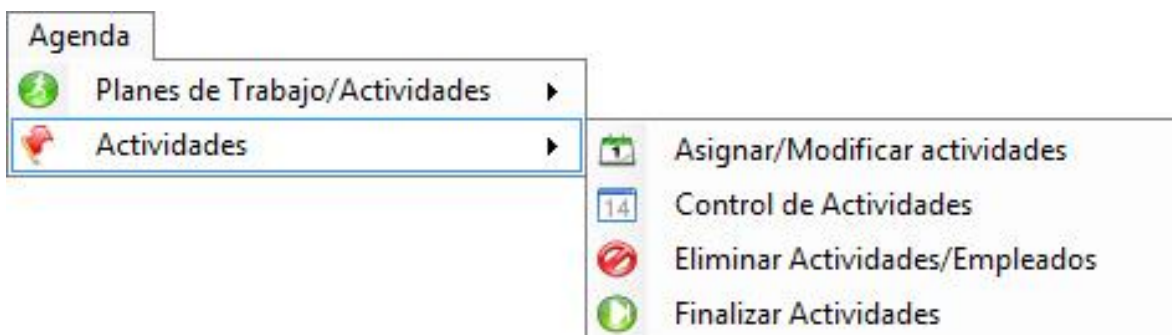


Figura 6.1.117 Sub-opciones de actividades

Sub-opción Asignar/Modificar Actividades

Permite asignar las actividades a los empleados con sus respectivas fechas de inicio de actividad, así como también brinda la posibilidad de modificar la fecha de las actividades en caso que no hayan iniciado.

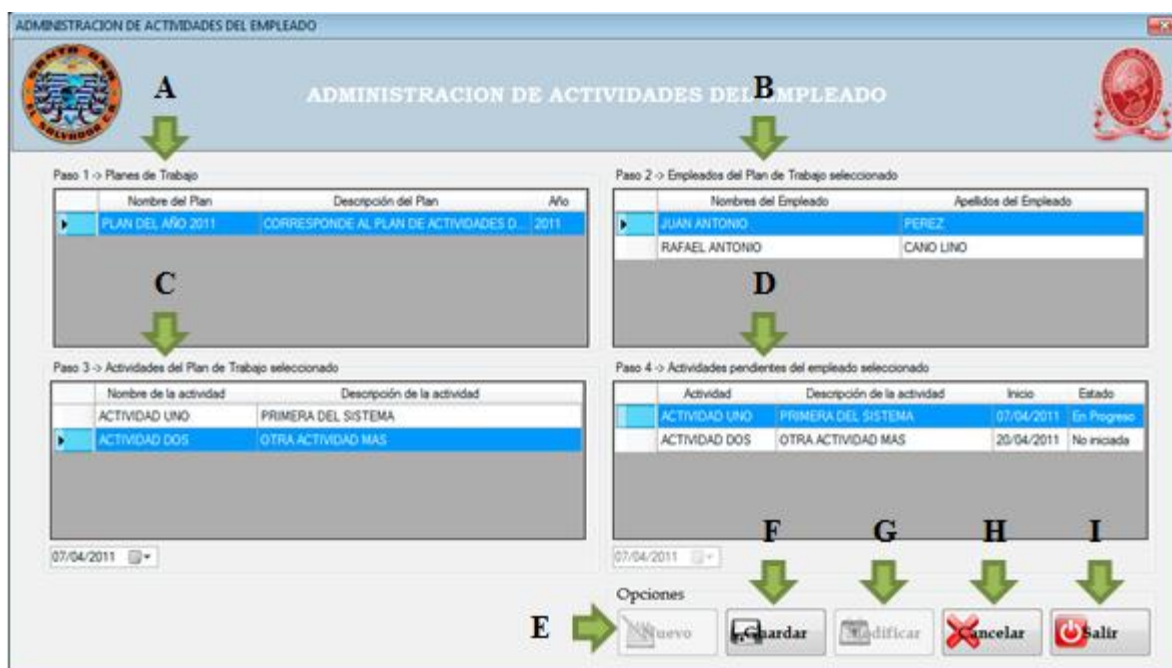


Figura 6.1.118 Asignación de actividades a los empleados

A. Muestra el listado de los planes de trabajo disponibles, debe de seleccionar un plan de trabajo para poder tener a disposición los empleados y actividades del plan de trabajo seleccionado para luego poder trabajar con estos.

B. Muestra el listado de los empleados del plan de trabajo seleccionado, debe seleccionar un empleado para poder asignarle actividades al mismo.

C. Muestra el listado de las actividades del plan de trabajo seleccionado, debe seleccionar una actividad para poder ser asignada al empleado seleccionado. Una vez ha seleccionado la actividad, observara que también hay una fecha la cual es la fecha de inicio de la actividad seleccionada, debe seleccionar la fecha en la cual iniciara la actividad.

D. Listado de las actividades no finalizadas del empleado seleccionado, estas actividades pueden ser pendientes o en progreso; este listado además de servir para mostrar las actividades del empleado nos da la posibilidad de poder modificar las fechas de inicio de dichas actividades, para eso se debe seleccionar la actividad a la cual se le desea modificar la fecha de inicio, luego observara que hay un indicador de fecha la cual indica la nueva fecha de inicio a modificar.

E. Al dar clic en la opción del menú aparecerá el formulario de la Figura 6.1.118 con el botón habilitado y los demás campos se encontraran inhabilitados, por lo que al hacer clic en este botón se habilitaran el listado los de planes de trabajo.

F. Representa al botón guardar (se puede hacer clic en este botón con la tecla Enter), que permite asignar actividades al empleado seleccionado, este botón solamente estará habilitado si existe algún plan de trabajo seleccionado, si este plan es del año actual o posterior y si esta seleccionado un empleado y una actividad. Otra funcionalidad de este botón es guardar los cambios cuando se ha modificado la fecha de inicio de una actividad asignada. Si hacemos clic en este botón, en el caso que estemos asignando una actividad nos preguntara lo siguiente:

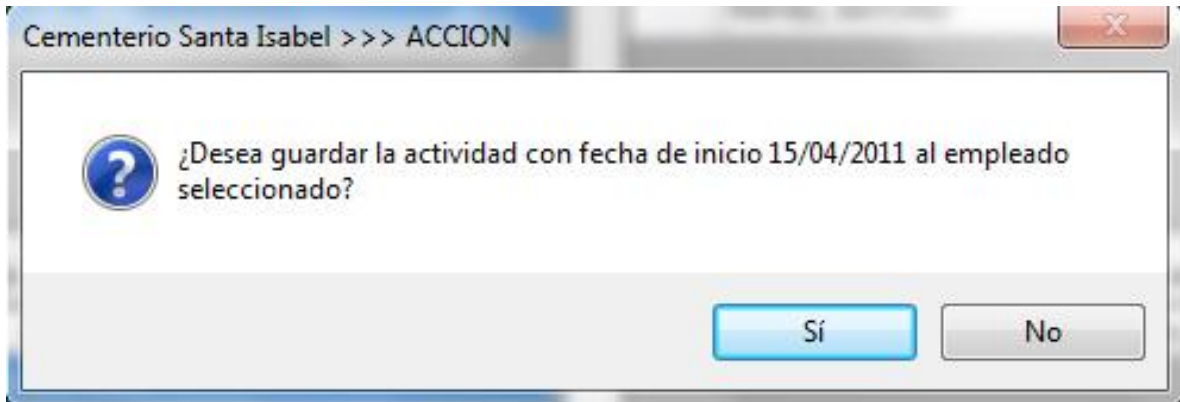


Figura 6.1.119 Mensaje preguntando por la asignación de actividades

Si respondemos afirmativamente a la pregunta anterior nos mostrara el siguiente mensaje de confirmación:

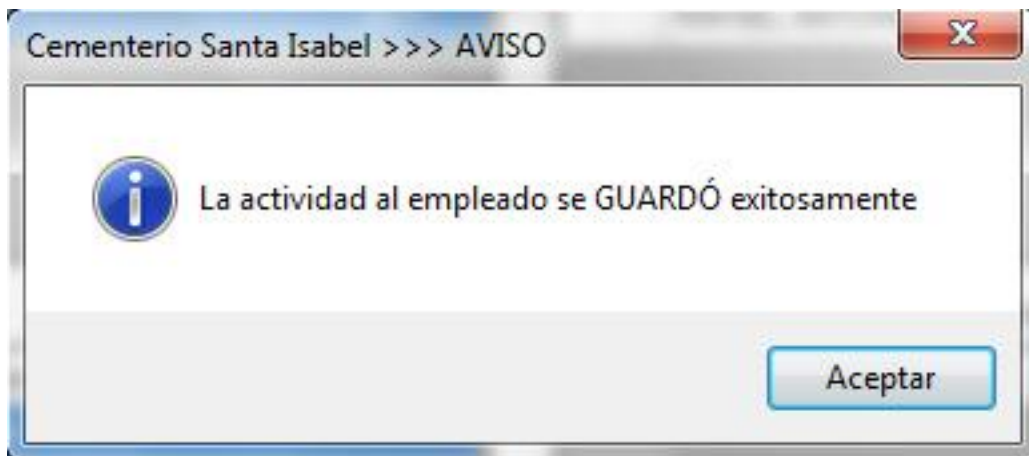


Figura 6.1.120 Mensaje confirmando asignación de actividad

Si sucediera el caso que estamos asignando una actividad repetida, es decir la misma actividad al mismo empleado en la misma fecha de inicio le mostrara el siguiente mensaje de error:

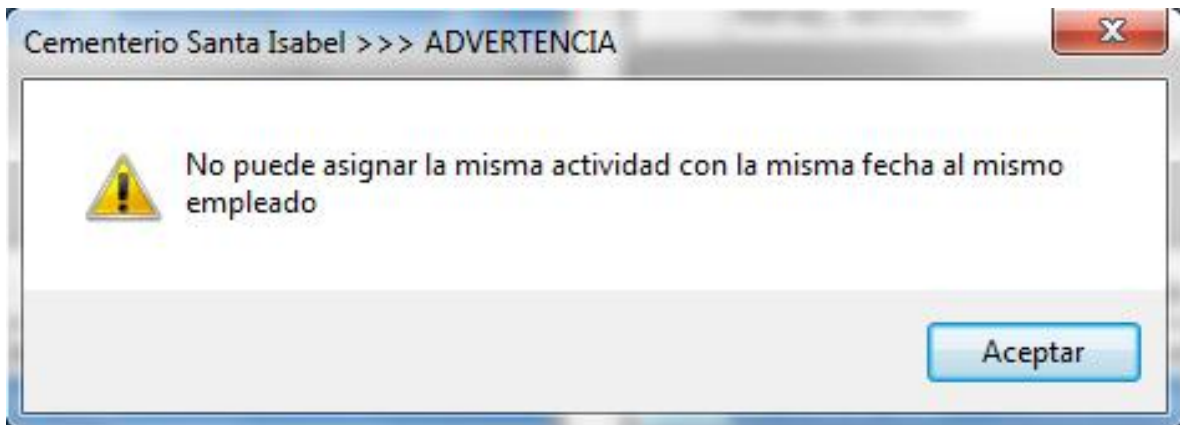


Figura 6.1.121 Mensaje de Error por asignación repetida

En el caso que estemos modificando la fecha de inicio de una actividad asignada, y dejemos la misma fecha de inicio como modificación, el sistema nos mostrara un mensaje de error por no realizar ninguna modificación, dicho mensaje es el siguiente:

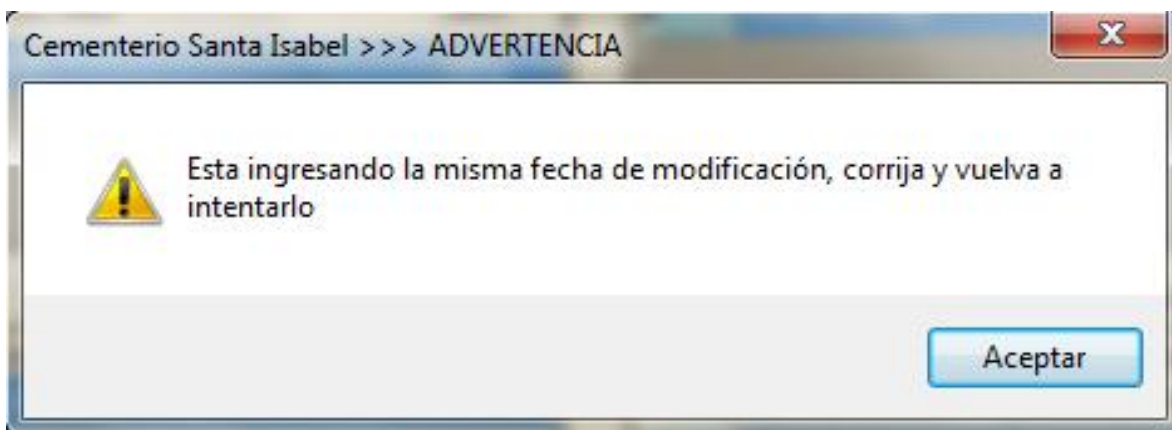


Figura 6.1.122 Mensaje de Error por no modificar fecha de inicio

En el caso que estemos modificando la fecha de inicio de una actividad asignada, nos preguntara si estamos seguros de modificar con el siguiente mensaje:

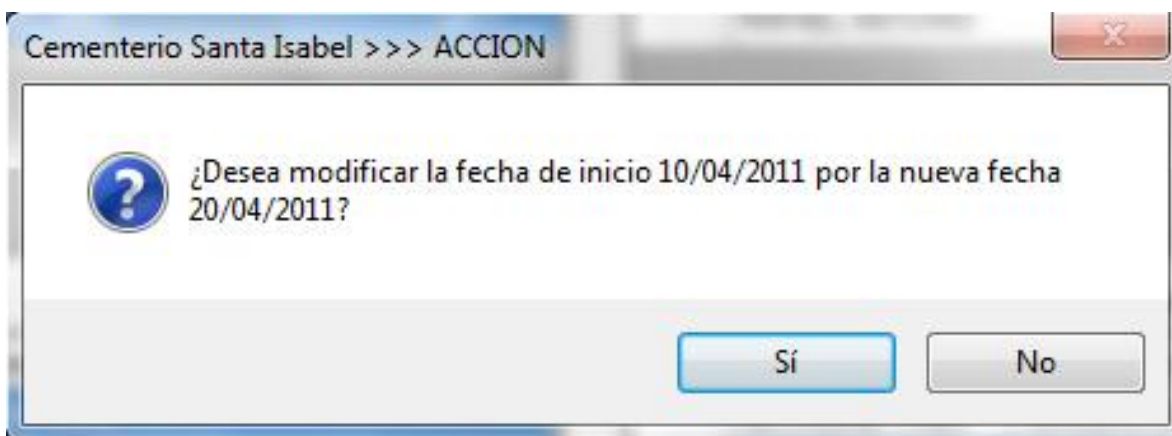


Figura 6.1.123 Mensaje preguntando por la modificación de actividad

Si decidimos modificar la fecha de inicio de la actividad, el sistema nos confirmara con el siguiente mensaje:

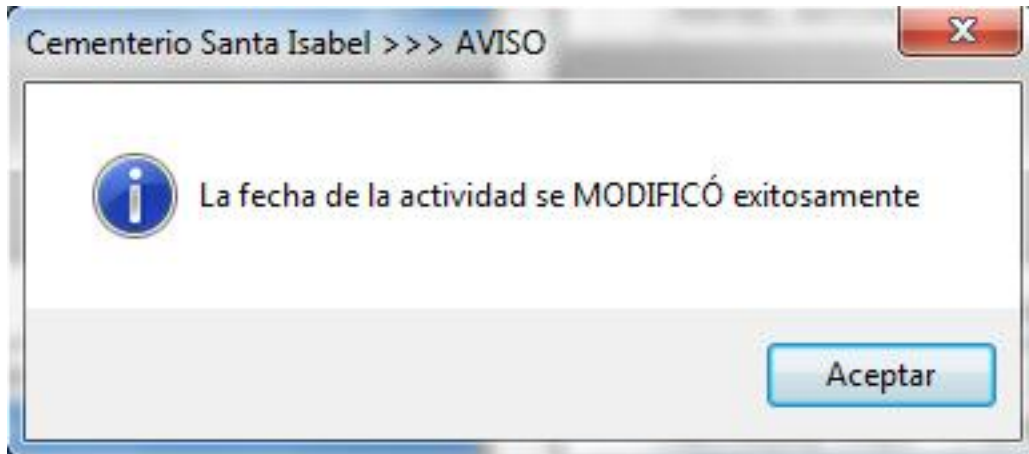


Figura 6.1.124 Mensaje confirmando la modificación de actividad

En el caso que la actividad a modificar se esté poniendo la misma fecha de inicio que la misma actividad al mismo empleado, es decir repitiendo la actividad el mismo día al mismo empleado, nos mostrara el mensaje de error correspondiente a la figura 6.1.121

G. Botón que permite habilitar la modificación de fecha de inicio de una actividad asignada; normalmente dicho botón se encuentra inhabilitado pero cuando es seleccionada una actividad que inicia como mínimo en la fecha actual del sistema, este botón se habilita y al hacer clic sobre este, brinda la posibilidad de modificar la fecha de inicio de la actividad seleccionada, a la vez que dicho botón se inhabilita y se habilita el botón de guardar.

H. Botón que nos permite cancelar cualquier acción que nos encontremos realizando; cuando hacemos clic en este botón nos muestra el mensaje de la figura 6.1.97

I. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.118, sin embargo debemos aclarar que si no hemos guardado un registro y presionamos este botón saldremos de la pantalla y los datos ingresados se perderán. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Sub-opción Control de Actividades

Permite tener un control sobre todas las actividades del sistema, como son actividades en progreso, realizadas, pendientes, etc.

Pestaña de Actividades del día Seleccionado

Permite mostrar las actividades que inician en el día seleccionado por el usuario.

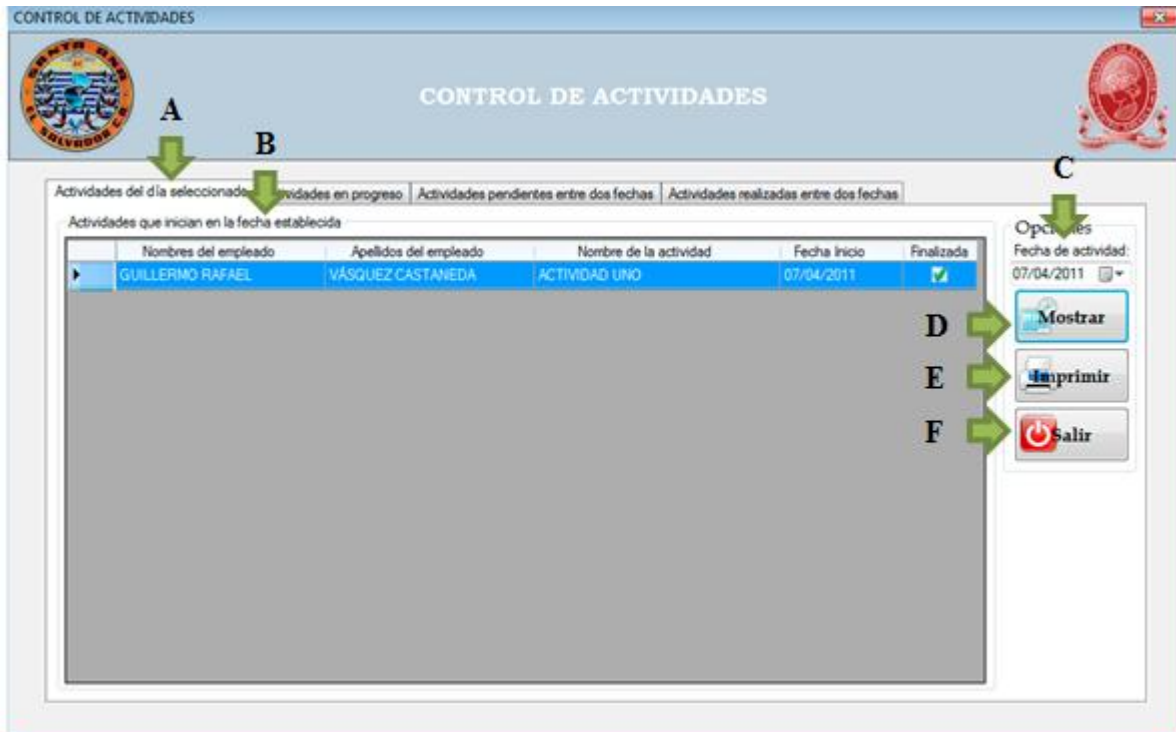


Figura 6.1.125 Actividades del día seleccionado

- A.** Esta pestaña indica que se está trabajando con las actividades del día seleccionado.
- B.** Muestra el listado de todas las actividades que tienen como fecha de inicio la fecha establecida por el indicador de fecha, además al final de cada actividad se indica si esta actividad ya finalizo o todavía no.
- C.** Permite establecer una fecha de inicio para poder ver todas las actividades que inician en la fecha que se establece en este control, por defecto este control tiene la fecha de inicio del día actual del sistema; la fecha mínima es el 1^o de Enero del año actual y la fecha máxima es el 31 de Diciembre del siguiente año.
- D.** Este botón permite mostrar todas las actividades que inician en la fecha establecida por el control anterior (se puede hacer clic en este botón con la tecla Enter); cuando se hace clic sobre este botón puede suceder que existan actividades en la fecha establecida, para lo cual el sistema nos avisa por medio del siguiente mensaje:

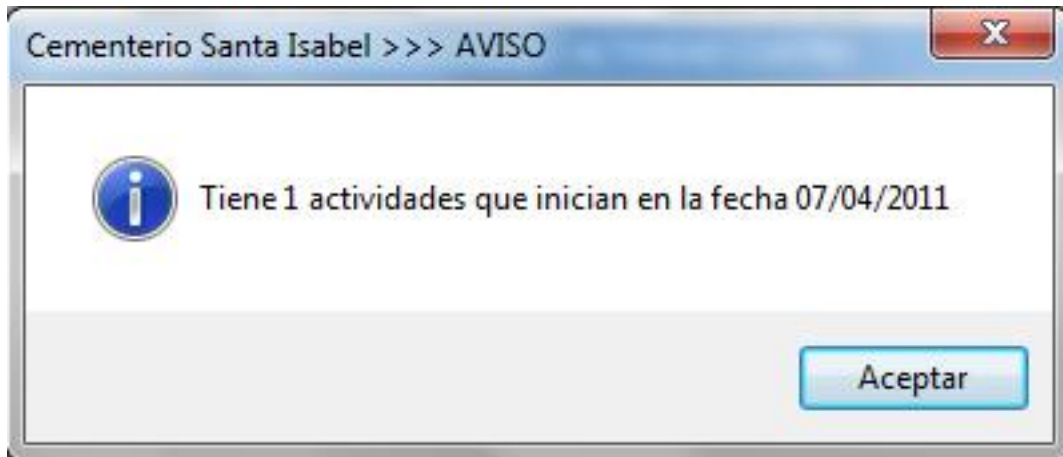


Figura 6.1.126 Mensaje indicando que hay actividades

Pero también puede suceder que no existan actividades en la fecha establecida, en este caso el sistema también nos avisa por medio del siguiente mensaje:

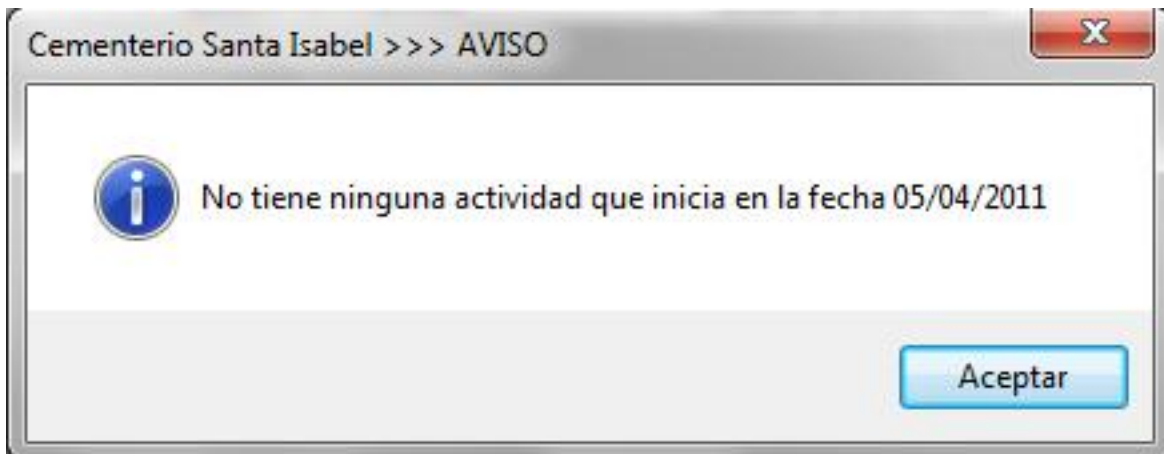


Figura 6.1.127 Mensaje indicando que no hay actividades

E. Botón que permite generar un reporte con las actividades que inician en la fecha establecida (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:



Figura 6.1.128 Reporte de las actividades que inician en la fecha establecida

E.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

F. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.125. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades en Progreso

Permite mostrar las actividades que se encuentran en progreso.

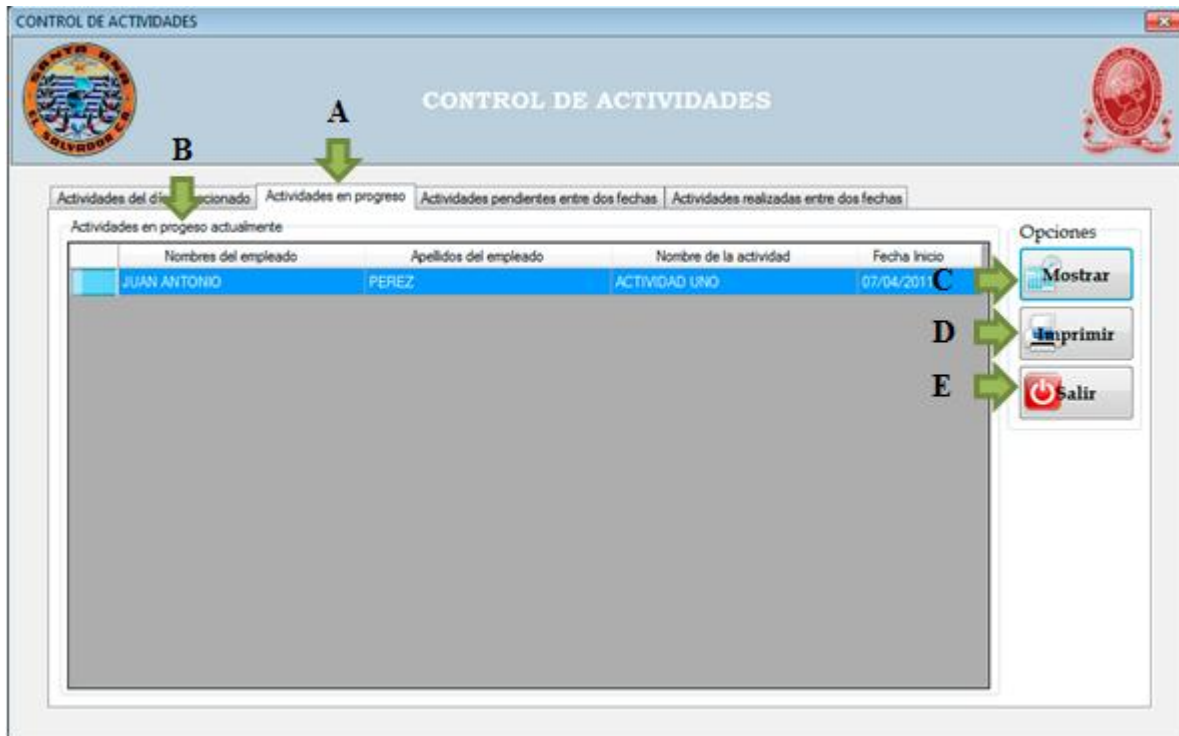


Figura 6.1.129 Actividades en progreso

- A.** Esta pestaña indica que se está trabajando con las actividades en progreso.
- B.** Muestra el listado de todas las actividades que se encuentran en progreso, es decir actividades que ya iniciaron pero no han sido finalizadas.
- C:** Este botón permite mostrar todas las actividades que se encuentran en progreso (se puede hacer clic en este botón con la tecla Enter); cuando se hace clic sobre este botón puede suceder que existan actividades en progreso, para lo cual el sistema nos avisa por medio del siguiente mensaje indicándonos además el número de actividades que se encuentran en esta condición:

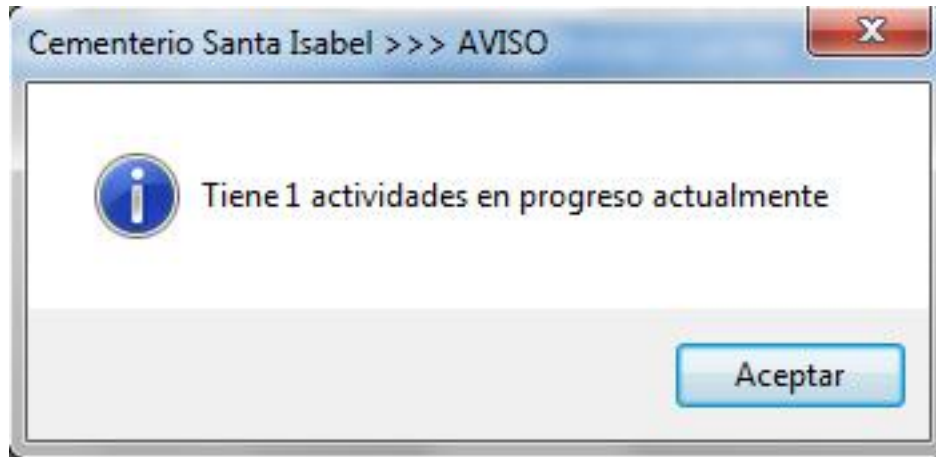


Figura 6.1.130 Mensaje indicando que hay actividades

Pero también puede suceder que no existan actividades en progreso, en este caso el sistema también nos avisa por medio del siguiente mensaje:

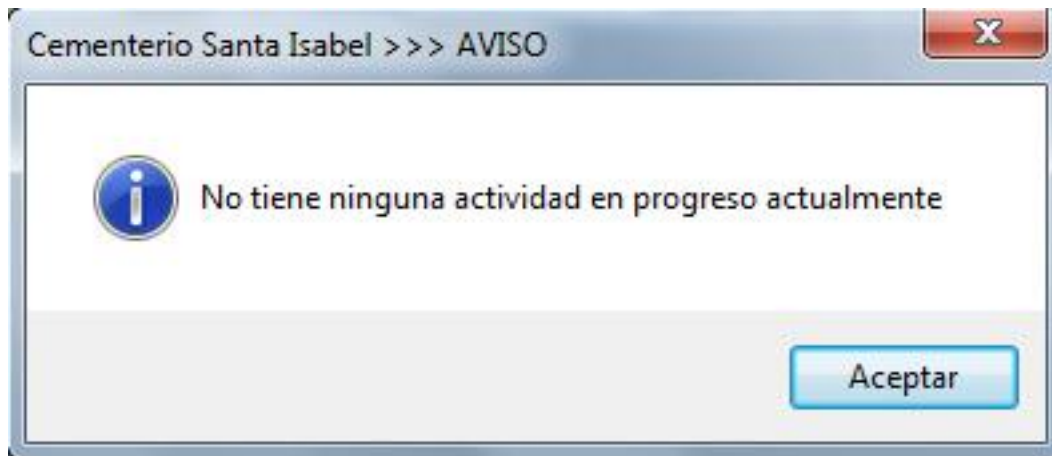


Figura 6.1.131 Mensaje indicando que no hay actividades

D. Botón que permite generar un reporte con las actividades en progreso (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:

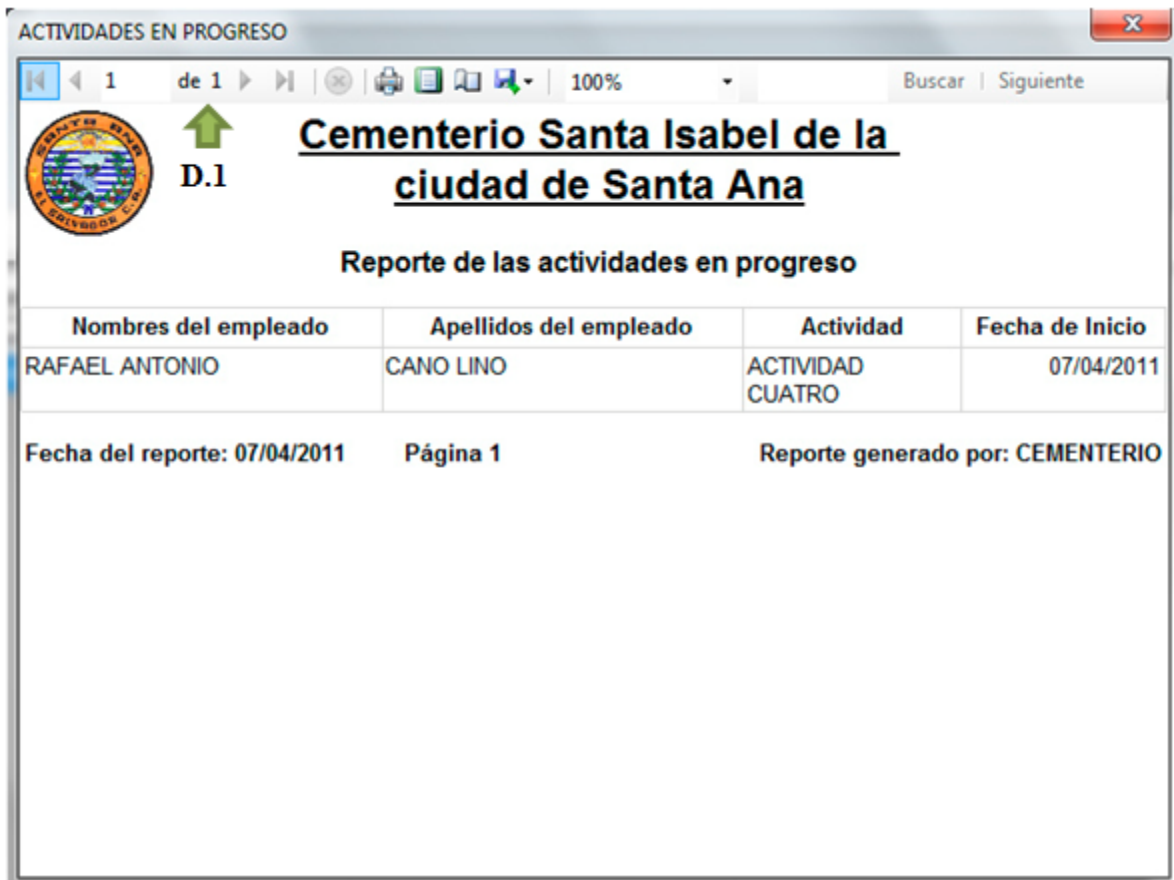


Figura 6.1.132 Reporte de las actividades en progreso

D.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

E. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.129. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades Pendientes entre dos fechas

Permite mostrar las actividades que se encuentran pendientes entre dos fechas establecidas por el usuario.

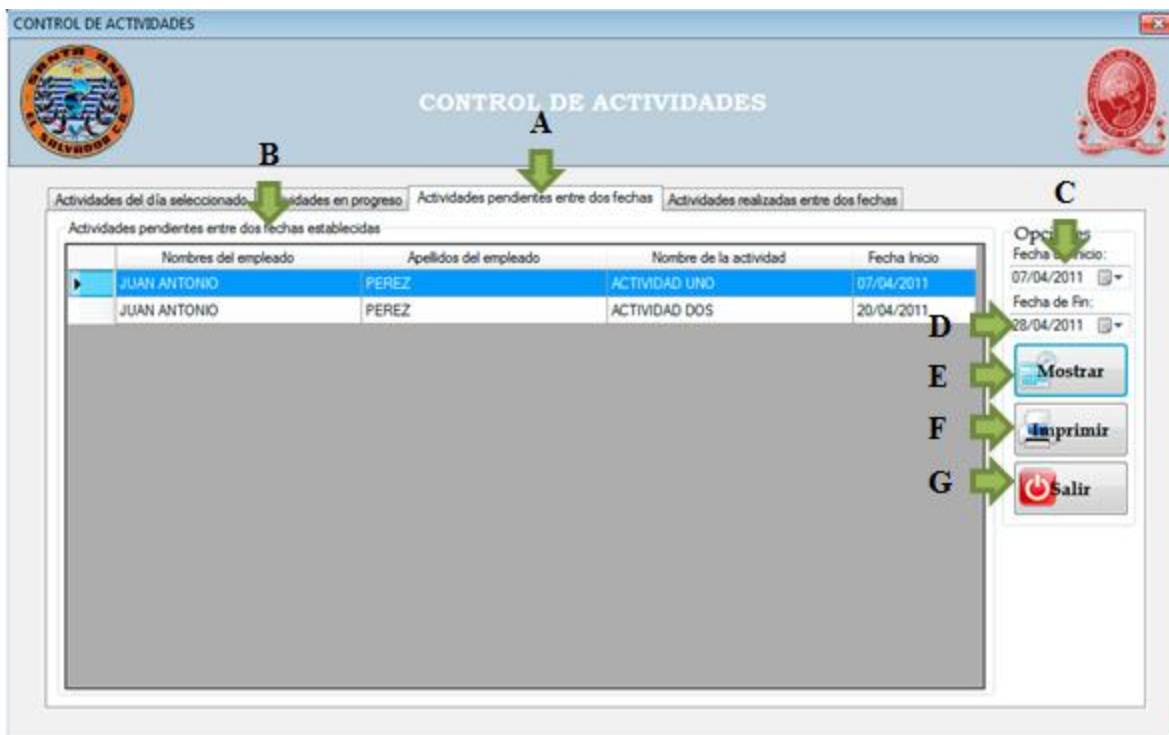


Figura 6.1.133 Actividades pendientes entre dos fechas

- A.** Esta pestaña indica que se está trabajando con las actividades pendientes entre dos fechas.
- B.** Muestra el listado de todas las actividades que están pendientes y se encuentran entre el intervalo de fechas definido.
- C.** Permite establecer una fecha inicial del intervalo de fechas; la fecha mínima es el día actual y la fecha máxima es el 31 de Diciembre del año siguiente. Esta fecha debe de ser menor o igual a la fecha final.
- D.** Permite establecer una fecha final del intervalo de fechas; la fecha mínima es el día actual y la fecha máxima es el 31 de Diciembre del año siguiente. Esta fecha debe de ser mayor o igual a la fecha inicial.
- E.** Este botón permite mostrar todas las actividades establecidas entre el intervalo de fechas establecido (se puede hacer clic en este botón con la tecla Enter); cuando se hace clic sobre este botón puede suceder que no se cumpla los requisitos de las fechas, es decir que la fecha inicial sea mayor a la final, en dado caso el sistema nos muestra el siguiente mensaje de error:

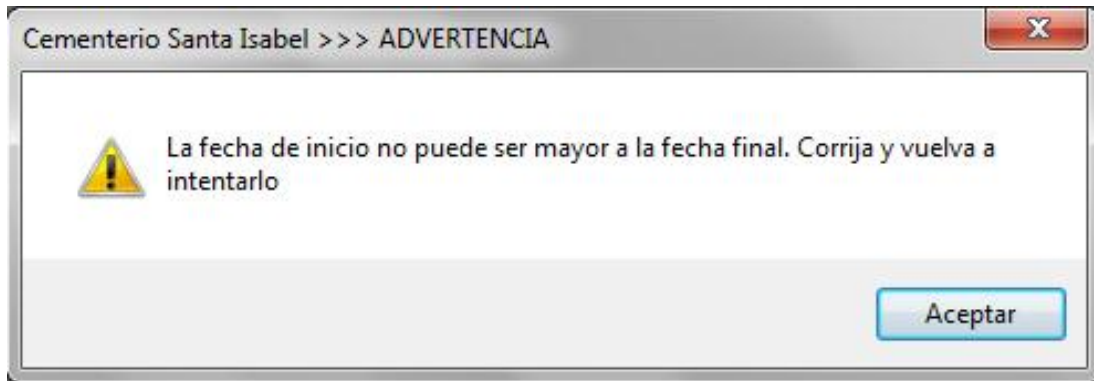


Figura 6.1.134 Mensaje de Error por fechas erróneas

En caso que las fechas cumplan el requisito y existan actividades entre el intervalo de fechas, el sistema muestra el siguiente mensaje indicándonos cuantas actividades se encuentran en dicho intervalo:

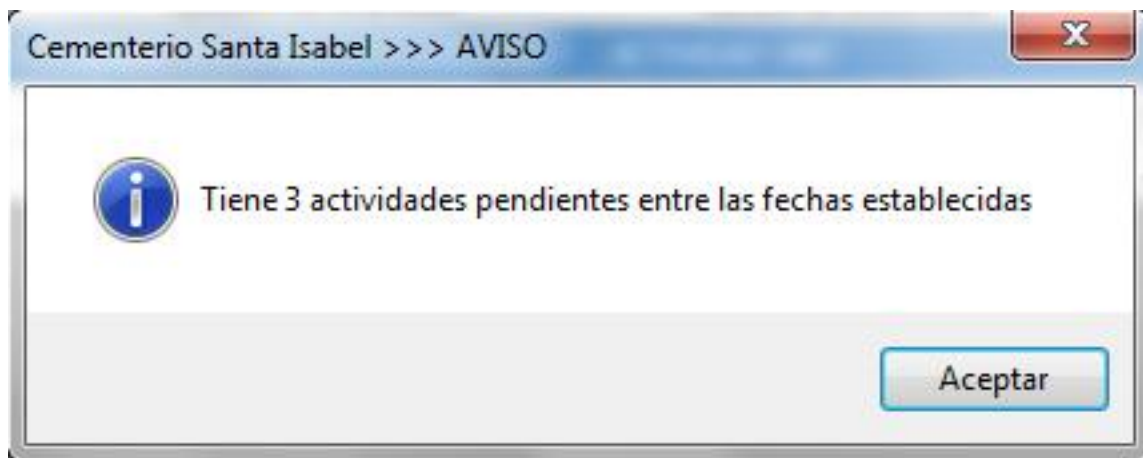


Figura 6.1.135 Mensaje indicando que hay actividades

Pero también puede suceder que no existan actividades entre el intervalo, en este caso el sistema también nos avisa por medio del siguiente mensaje:

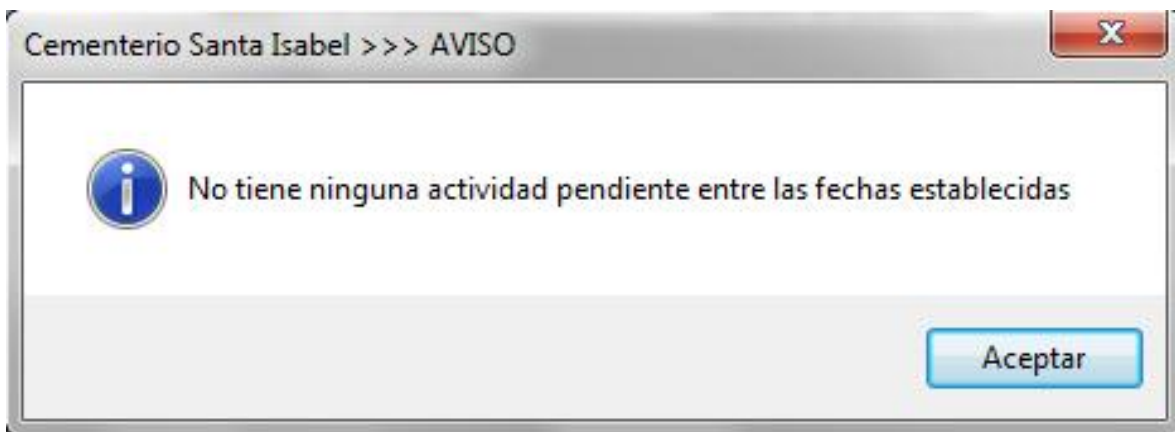


Figura 6.1.136 Mensaje indicando que no hay actividades

F. Botón que permite generar un reporte con las actividades pendientes entre las fechas establecidas (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:

ACTIVIDADES PENDIENTES

1 de 1 | 100% | Buscar | Siguiente

Cementerio Santa Isabel de la ciudad de Santa Ana

Reporte de las actividades pendientes entre las fechas 07/04/2011 y 02/05/2011

Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio
RODOLFO MANUEL	SOLIS	ACTIVIDAD UNO	15/04/2011
RODOLFO MANUEL	SOLIS	ACTIVIDAD UNO	20/04/2011
JUAN ANTONIO	PEREZ	ACTIVIDAD DOS	20/04/2011

Fecha del reporte: 07/04/2011 Página 1 Reporte generado por: CEMENTERIO

Figura 6.1.137 Reporte de las actividades pendientes

F.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.

6) Búsqueda dentro del reporte.

G. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.133. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades realizadas entre dos fechas

Permite mostrar las actividades que se han realizado entre dos fechas establecidas por el usuario.

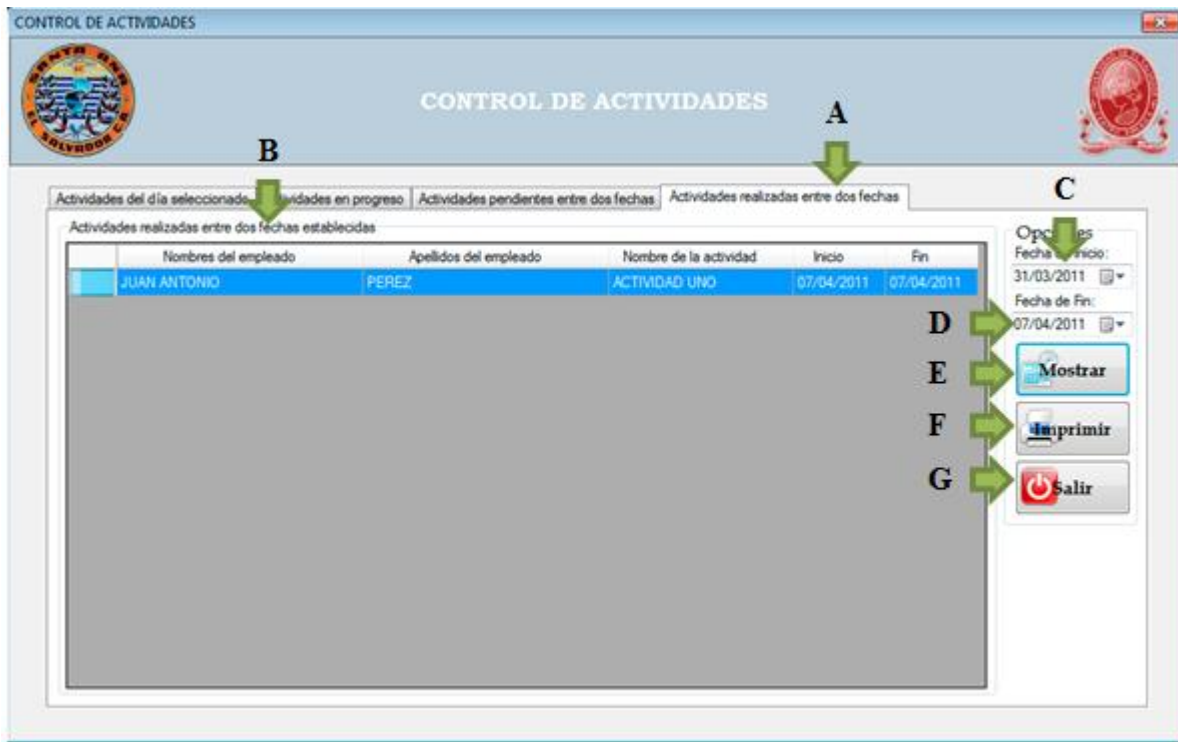


Figura 6.1.138 Actividades realizadas entre dos fechas

A. Esta pestaña indica que se está trabajando con las actividades realizadas entre dos fechas.

B. Muestra el listado de todas las actividades que se han realizado (es decir ya finalizadas) y se encuentran entre el intervalo de fechas definido.

C. Permite establecer una fecha inicial del intervalo de fechas; la fecha mínima es el 1º de Enero del año actual y la fecha máxima es el día actual. Esta fecha debe de ser menor o igual a la fecha final.

D. Permite establecer una fecha final del intervalo de fechas; la fecha mínima es el 1º de Enero del año actual y la fecha máxima es el día actual. Esta fecha debe de ser mayor o igual a la fecha inicial.

E. Este botón permite mostrar todas las actividades establecidas entre el intervalo de fechas establecido (se puede hacer clic en este botón con la tecla Enter); cuando se hace clic sobre este botón puede suceder que no se cumpla los requisitos de las fechas, es decir que la fecha inicial sea mayor a la final, en dado caso el sistema muestra el mensaje de error correspondiente a la figura 6.1.134

En caso que las fechas cumplan el requisito y existan actividades entre el intervalo de fechas, el sistema muestra el siguiente mensaje indicándonos cuantas actividades se encuentran en dicho intervalo:

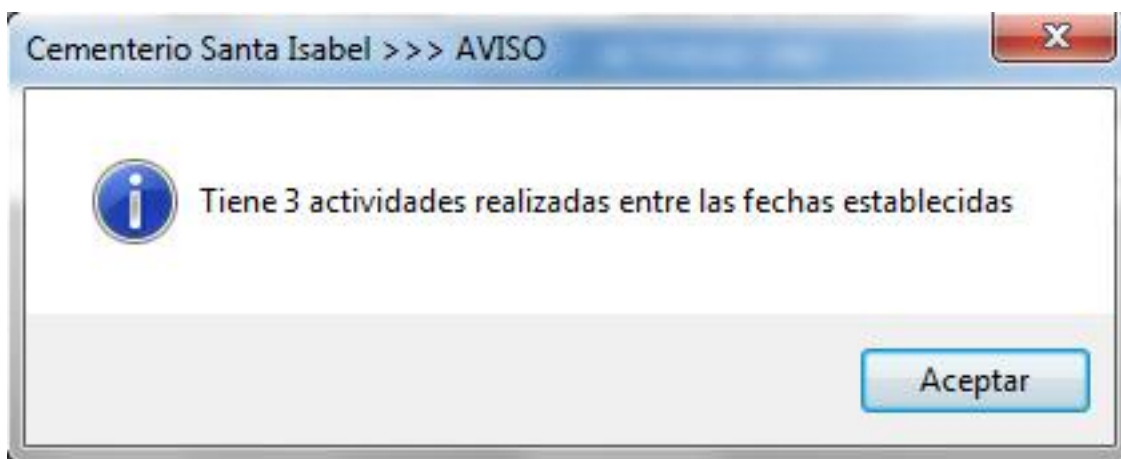


Figura 6.1.139 Mensaje indicando que hay actividades

Pero también puede suceder que no existan actividades entre el intervalo, en este caso el sistema también nos avisa por medio del siguiente mensaje:

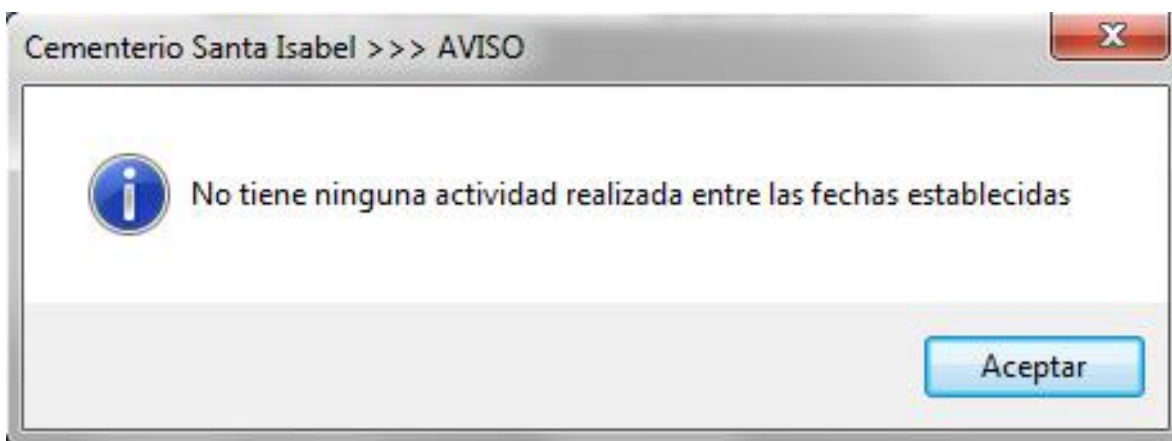
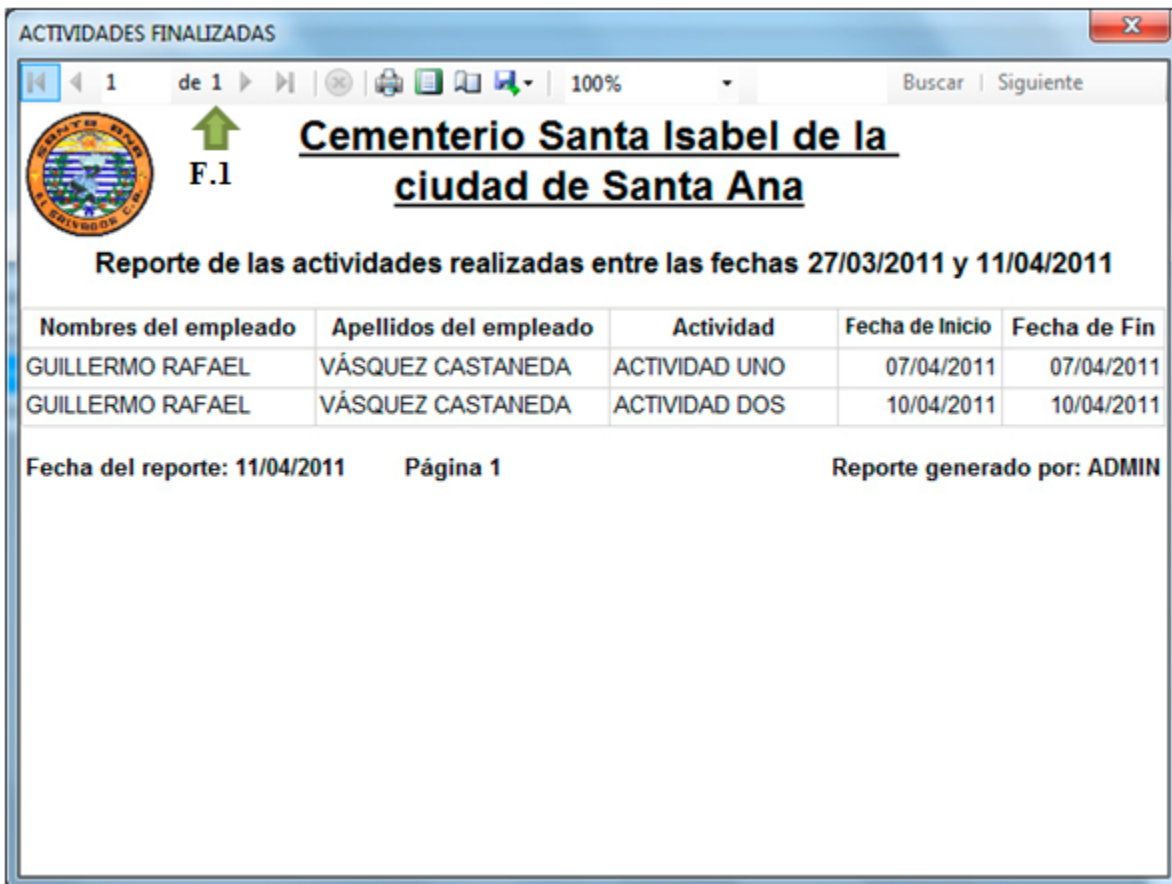


Figura 6.1.140 Mensaje indicando que no hay actividades

F. Botón que permite generar un reporte con las actividades realizadas entre las fechas establecidas (habilitado solo en el caso que existan actividades), dicho reporte puede ser impreso o ser exportado a documento PDF; para tal propósito cuando se hace clic en

dicho botón muestra el siguiente formulario donde nos ofrece las opciones antes mencionadas:



Nombres del empleado	Apellidos del empleado	Actividad	Fecha de Inicio	Fecha de Fin
GUILLERMO RAFAEL	VÁSQUEZ CASTANEDA	ACTIVIDAD UNO	07/04/2011	07/04/2011
GUILLERMO RAFAEL	VÁSQUEZ CASTANEDA	ACTIVIDAD DOS	10/04/2011	10/04/2011

Fecha del reporte: 11/04/2011 Página 1 Reporte generado por: ADMIN

Figura 6.1.141 Reporte de las actividades realizadas

F.1 Esta barra nos facilita un variado repertorio de opciones sobre el reporte, las cuales son:

- 1) Navegar entre las diferentes páginas del reporte.
- 2) Detener la carga del reporte, esto significa que cuando se están cargando los datos podemos evitar eso para volver a la ventana anterior.
- 3) Imprimir el reporte así como también poder configurar la impresión, teniendo la posibilidad de tener una vista previa del reporte antes de ser impreso.
- 4) Manejar el acercamiento de la vista del reporte, lo cual permite acercar o alejar la vista del reporte.
- 5) Exportar el reporte a documento PDF, el cual puede ser posteriormente impreso.
- 6) Búsqueda dentro del reporte.

G. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.138. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Sub-opción Eliminar Actividades/Empleados

Permite eliminar actividades del cementerio, quitarlas de los planes de trabajo, así como también quitar empleados de los planes de trabajo y cancelar actividades de los empleados.

Pestaña Actividades

Permite eliminar actividades, siempre y cuando estas actividades no estén pendientes de ser realizadas por algún empleado.

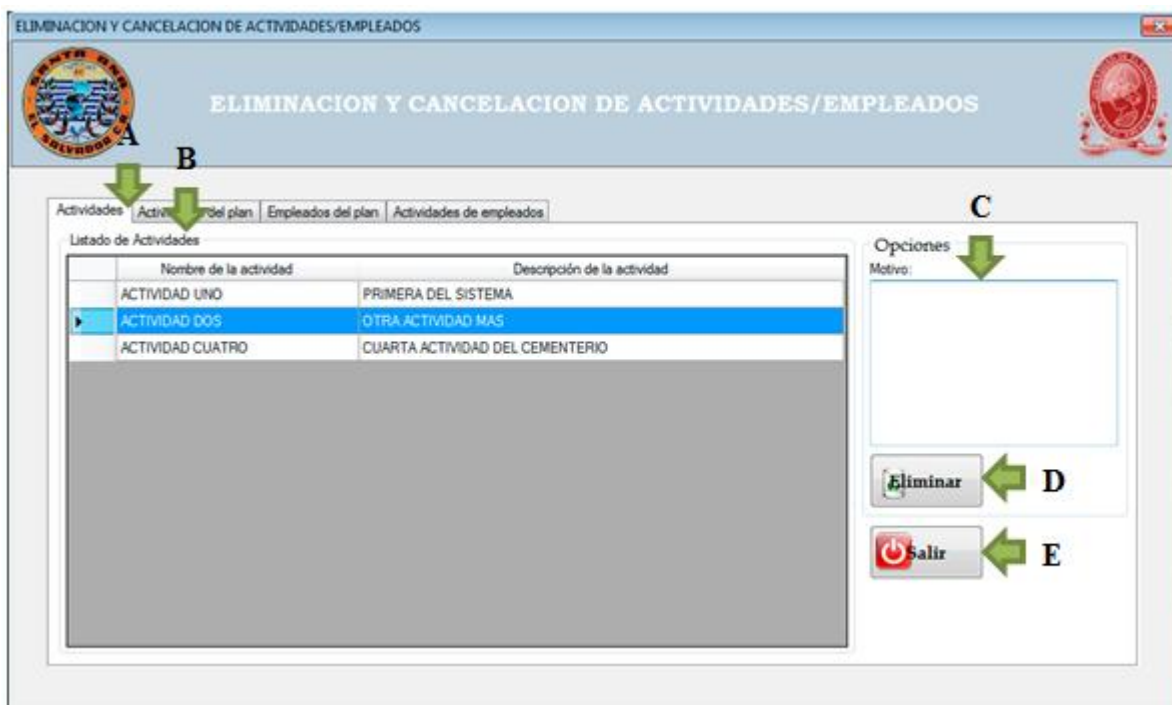


Figura 6.1.142 Eliminación de actividades

- A.** Esta pestaña indica que se está trabajando con la eliminación de actividades.
- B.** Muestra el listado de todas las actividades que están registradas en el sistema. Debe seleccionar la actividad que desea eliminar para luego poder ser eliminada.
- C.** En este campo debe de escribir el motivo por el cual elimina la actividad seleccionada, es obligatorio escribir el motivo de lo contrario el sistema impedirá que la actividad sea eliminada; cabe destacar que los caracteres especiales no están permitidos en este campo, además el número máximo es de 150 caracteres.

D. Este botón permite eliminar la actividad seleccionada (se puede hacer clic en este botón con la tecla Enter), siempre y cuando se haya ingresado un motivo para eliminar dicha actividad a la vez que dicha actividad no esté pendiente de realizar por algún empleado. Si se da el caso que no se ha escrito un motivo para la eliminación el sistema muestra el siguiente mensaje de error:

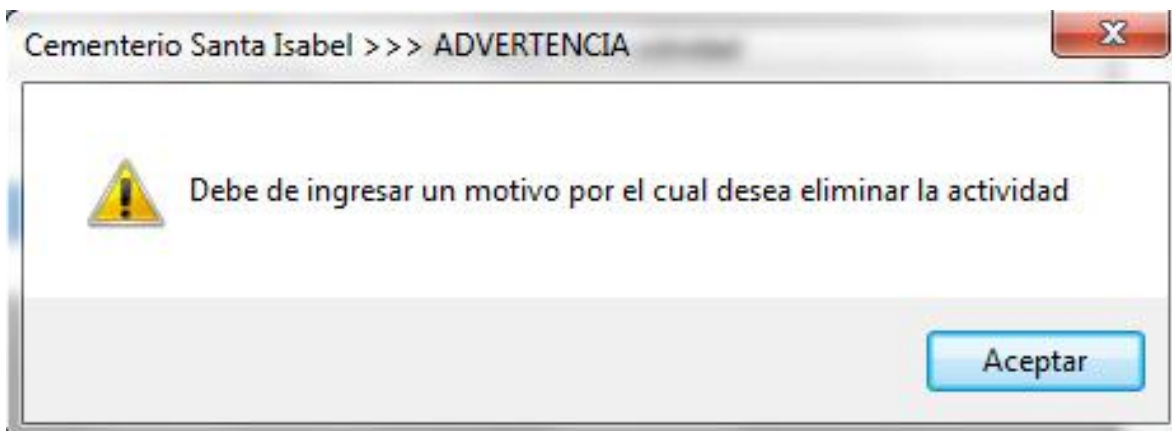


Figura 6.1.143 Mensaje de Error por no escribir un motivo

También puede suceder que la actividad este pendiente de realizar por algún empleado en cuyo caso el sistema nos avisa que no es posible eliminar la actividad por medio del siguiente mensaje:

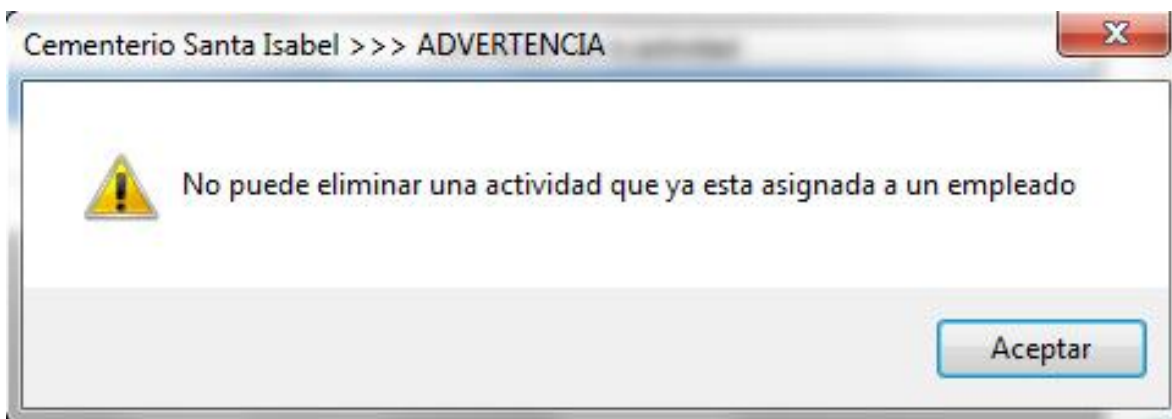


Figura 6.1.144 Mensaje de Error porque la actividad está pendiente

Si se ha escrito un motivo para la eliminación a la vez que la actividad no está pendiente de ser realizada, el sistema nos preguntara si estamos seguros de eliminar la actividad por medio del siguiente mensaje:

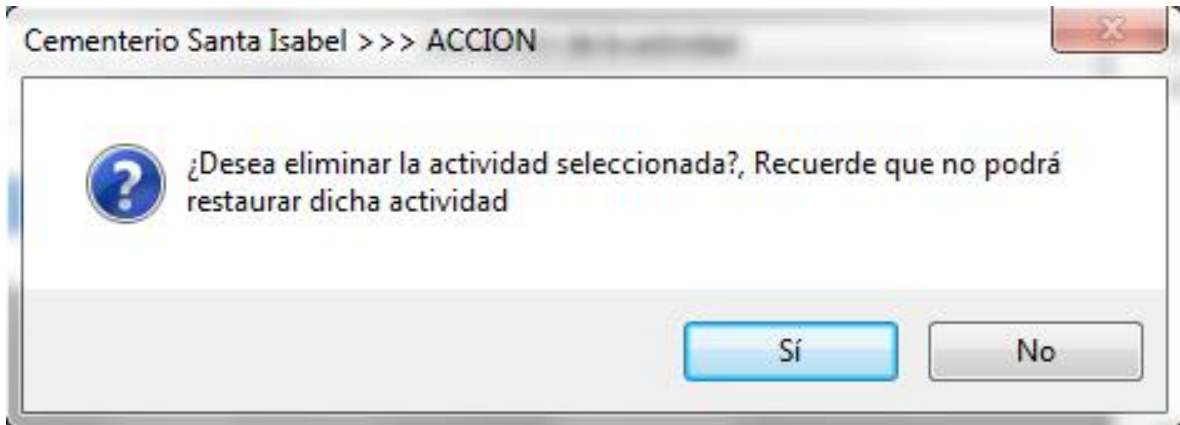


Figura 6.1.145 Mensaje preguntando por la eliminación de actividad

En caso que respondamos afirmativamente el sistema nos mostrara el siguiente mensaje de confirmación:

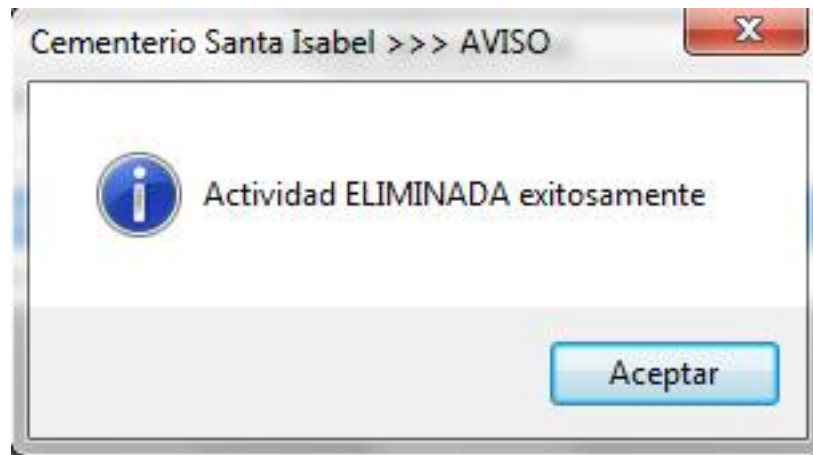


Figura 6.1.146 Mensaje confirmando la eliminación

E. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.142. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades del Plan

Permite eliminar las actividades del plan, en pocas palabras permite quitarlas del plan pero no las elimina del sistema, siempre estarán disponibles.

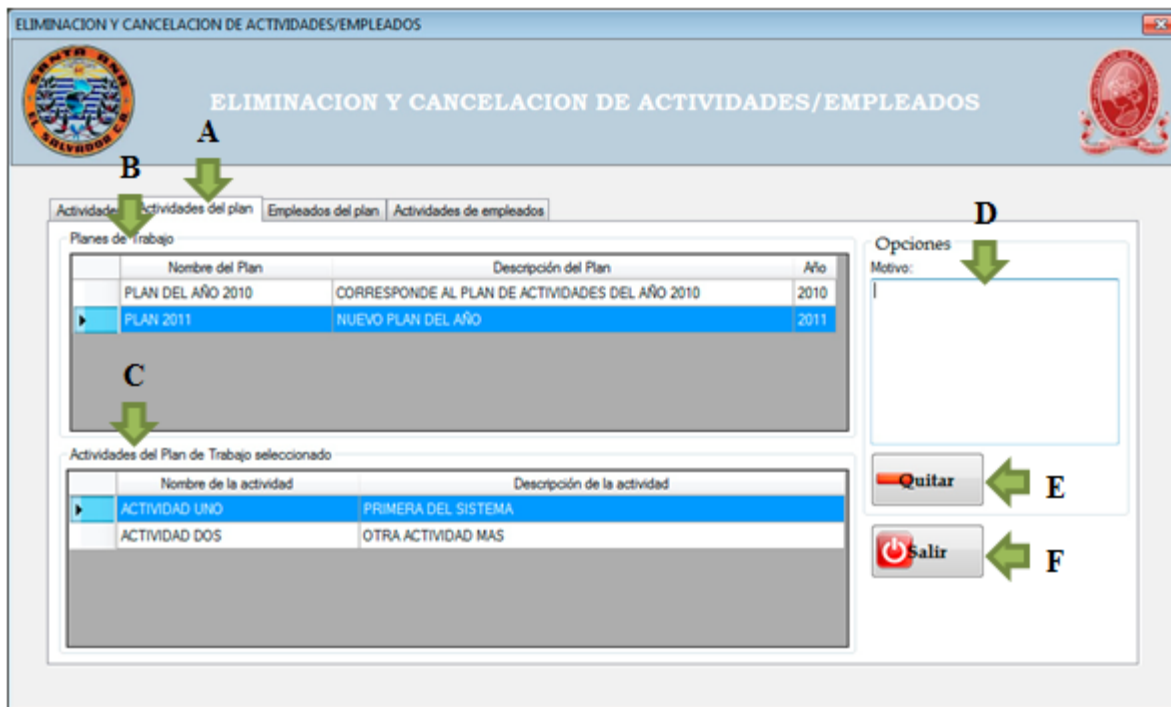


Figura 6.1.147 Eliminación de actividades del plan

- A.** Esta pestaña indica que se está trabajando con la eliminación de actividades del plan de trabajo.
- B.** Muestra el listado de todos los planes de trabajo que contienen las actividades a quitar. Debe seleccionar un plan para poder acceder a sus actividades respectivas.
- C.** Muestra el listado de todas las actividades que pertenecen al plan de trabajo seleccionado. Solamente podrán quitarse aquellas actividades que pertenezcan al plan de trabajo del año actual o siguiente. Debe seleccionar la actividad que desea quitar del plan de trabajo seleccionado.
- D.** En este campo debe de escribir el motivo por el cual quita la actividad seleccionada, es obligatorio escribir el motivo de lo contrario el sistema impedirá que la actividad sea quitada; cabe destacar que los caracteres especiales no están permitidos en este campo, además el número máximo es de 150 caracteres.
- E.** Este botón permite quitar la actividad seleccionada (se puede hacer clic en este botón con la tecla Enter), siempre y cuando se haya ingresado un motivo para quitar dicha actividad a la vez que dicha actividad no esté pendiente de realizar por algún empleado. Si se da el caso que no se ha escrito un motivo para la eliminación el sistema muestra el siguiente mensaje de error:

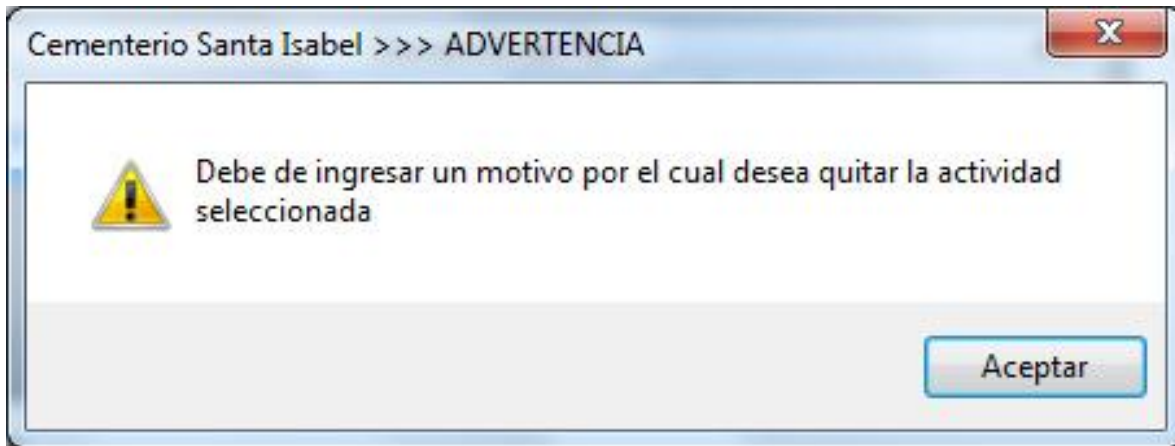


Figura 6.1.148 Mensaje de Error por no escribir un motivo

También puede suceder que la actividad este pendiente de realizar por algún empleado en cuyo caso el sistema nos avisa que no es posible eliminar la actividad por medio del mensaje correspondiente a la figura 6.1.144

Si se ha escrito un motivo para la eliminación a la vez que la actividad no está pendiente de ser realizada, el sistema nos preguntara si estamos seguros de quitar la actividad por medio del siguiente mensaje:

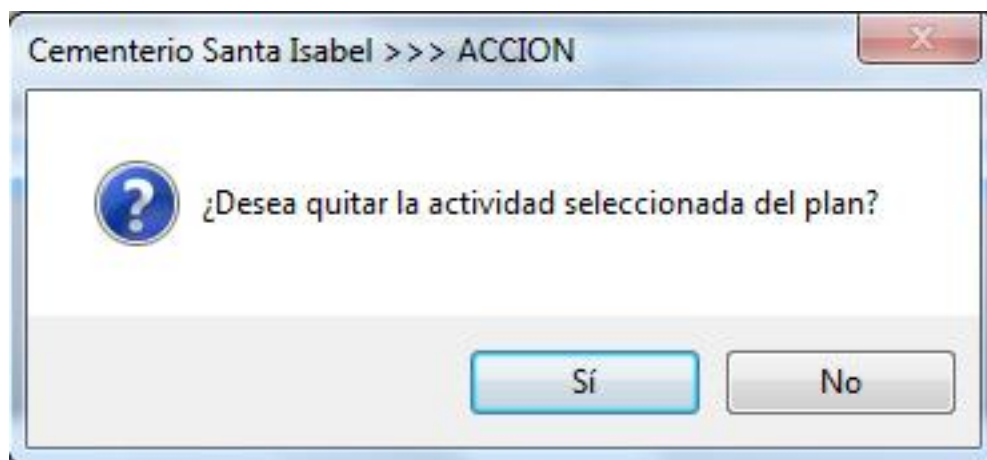


Figura 6.1.149 Mensaje preguntando por la eliminación de actividad

En caso que respondamos afirmativamente el sistema nos mostrara el siguiente mensaje de confirmación:

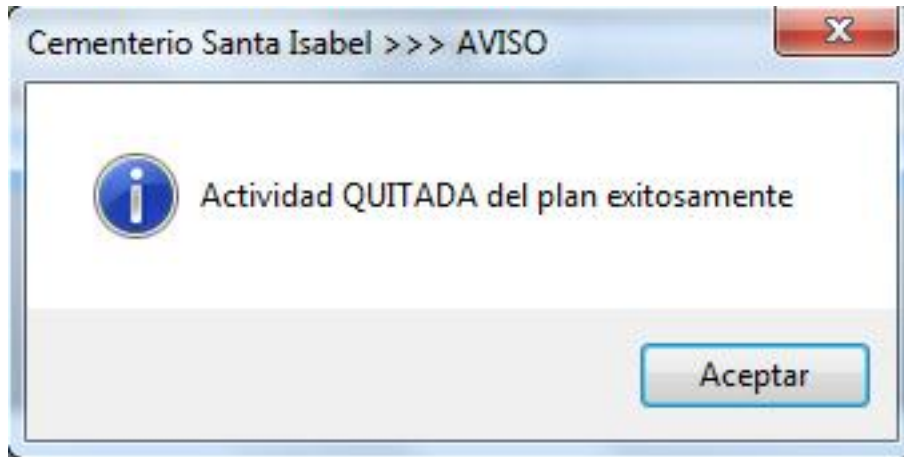


Figura 6.1.150 Mensaje confirmando la eliminación

F. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.147. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Empleados del Plan

Permite eliminar los empleados del plan, en pocas palabras permite quitarlos del plan pero no los elimina del sistema, siempre estarán disponibles.

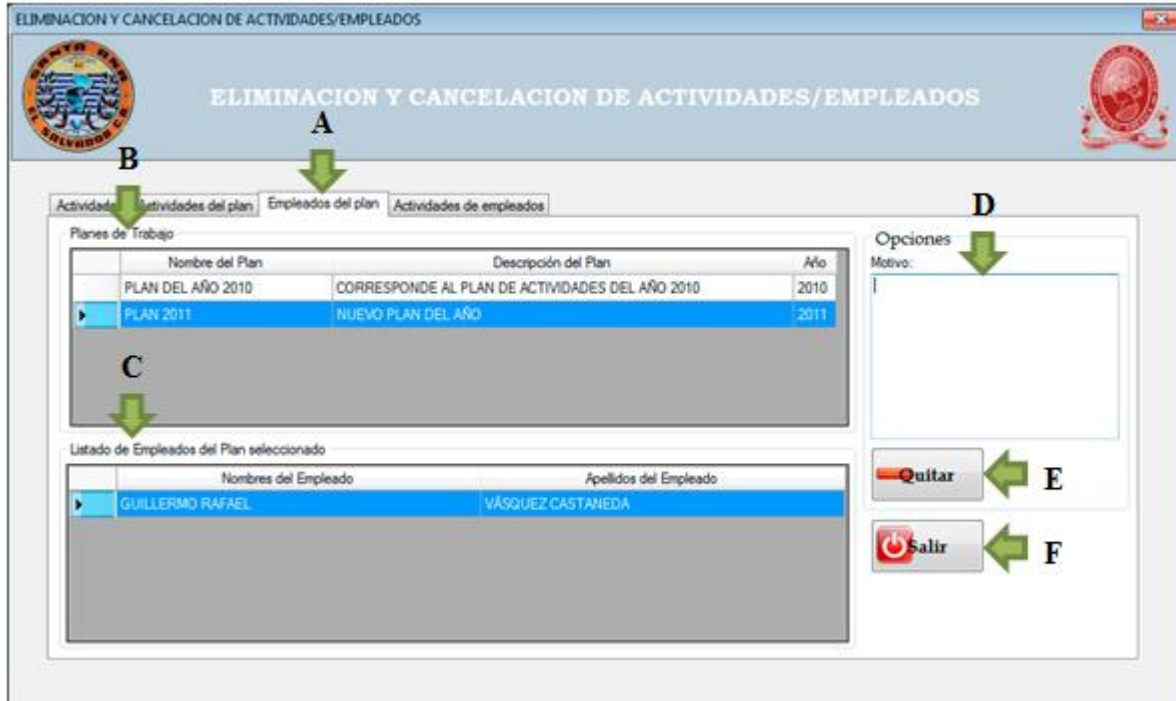


Figura 6.1.151 Eliminación de empleados del plan

- A.** Esta pestaña indica que se está trabajando con la eliminación de empleados del plan de trabajo.
- B.** Muestra el listado de todos los planes de trabajo que contienen los empleados a quitar. Debe seleccionar un plan para poder acceder a sus empleados respectivos.
- C.** Muestra el listado de todos los empleados que pertenecen al plan de trabajo seleccionado. Solamente podrán quitarse aquellos empleados que pertenezcan al plan de trabajo del año actual o siguiente. Debe seleccionar el empleado que desea quitar del plan de trabajo seleccionado.
- D.** En este campo debe de escribir el motivo por el cual quita al empleado seleccionado, es obligatorio escribir el motivo de lo contrario el sistema impedirá que el empleado sea quitado; cabe destacar que los caracteres especiales no están permitidos en este campo, además el número máximo es de 150 caracteres.
- E.** Este botón permite quitar al empleado seleccionado (se puede hacer clic en este botón con la tecla Enter), siempre y cuando se haya ingresado un motivo para quitar dicho empleado a la vez que este no esté pendiente de realizar ninguna actividad. Si se da el caso que no se ha escrito un motivo para la eliminación el sistema muestra el siguiente mensaje de error:

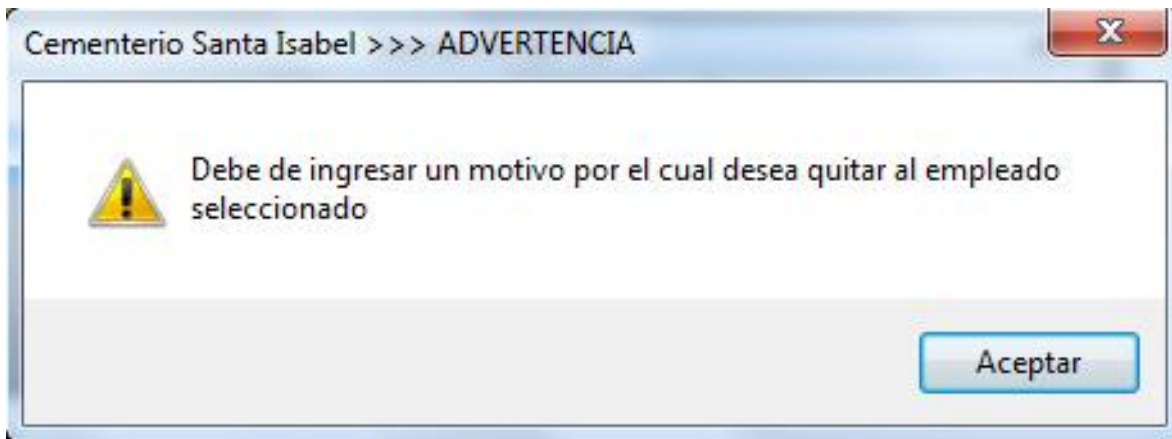


Figura 6.1.152 Mensaje de Error por no escribir un motivo

También puede suceder que el empleado tenga actividades pendientes de realizar en cuyo caso el sistema nos avisa que no es posible eliminar al empleado por medio del siguiente mensaje:

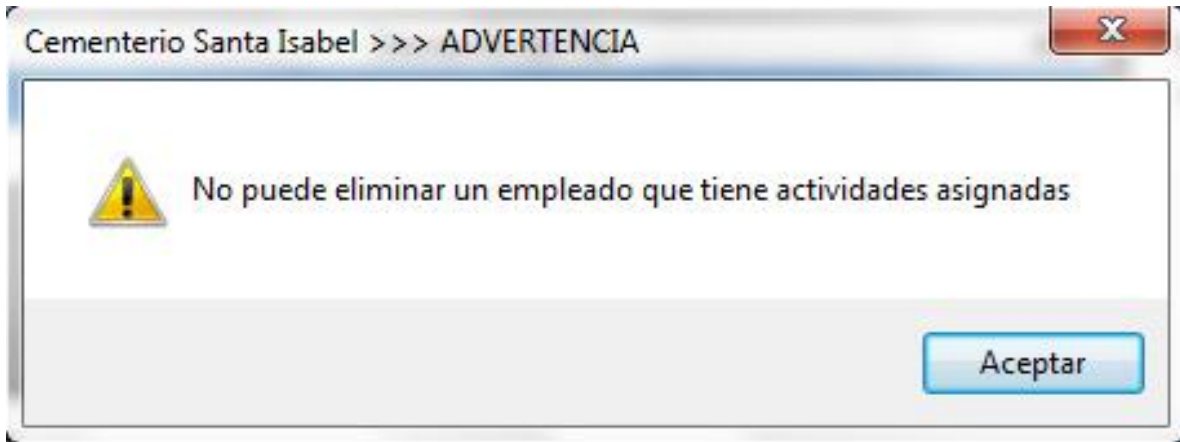


Figura 6.1.153 Mensaje de Error porque el empleado tiene actividades pendientes

Si se ha escrito un motivo para la eliminación a la vez que el empleado no tiene actividades pendientes, el sistema nos preguntara si estamos seguros de quitar el empleado por medio del siguiente mensaje:

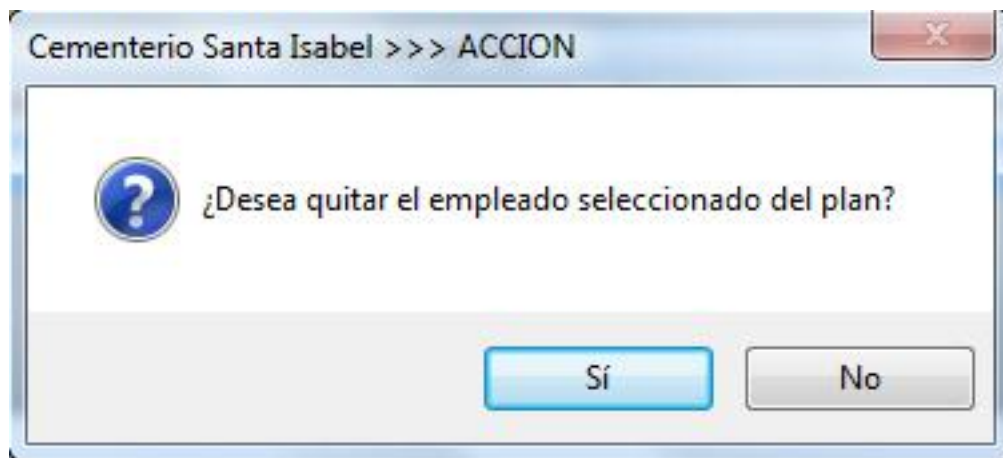


Figura 6.1.154 Mensaje preguntando por la eliminación del empleado

En caso que respondamos afirmativamente el sistema nos mostrara el siguiente mensaje de confirmación:

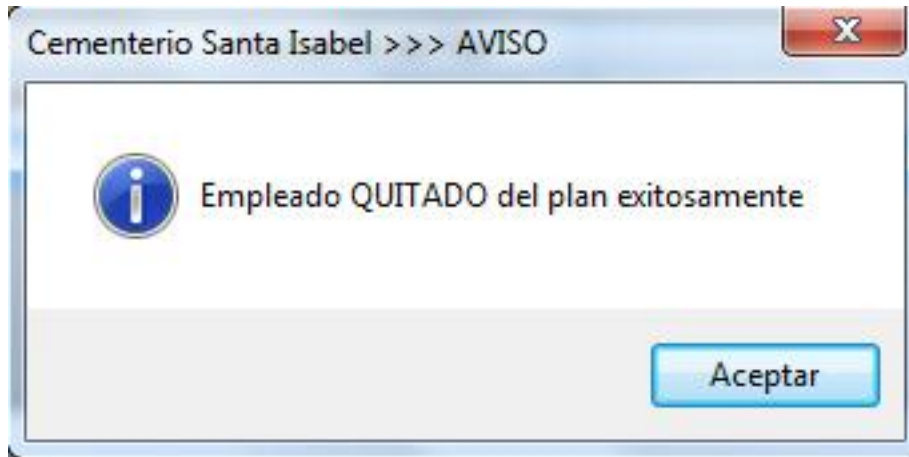


Figura 6.1.155 Mensaje confirmando la eliminación

F. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.151. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Pestaña de Actividades de empleados

Permite cancelar las actividades de los empleados que están pendientes, permitiendo la cancelación de las mismas hasta el mismo día en que inician, pero no cuando tienen más de un día de haber iniciado.

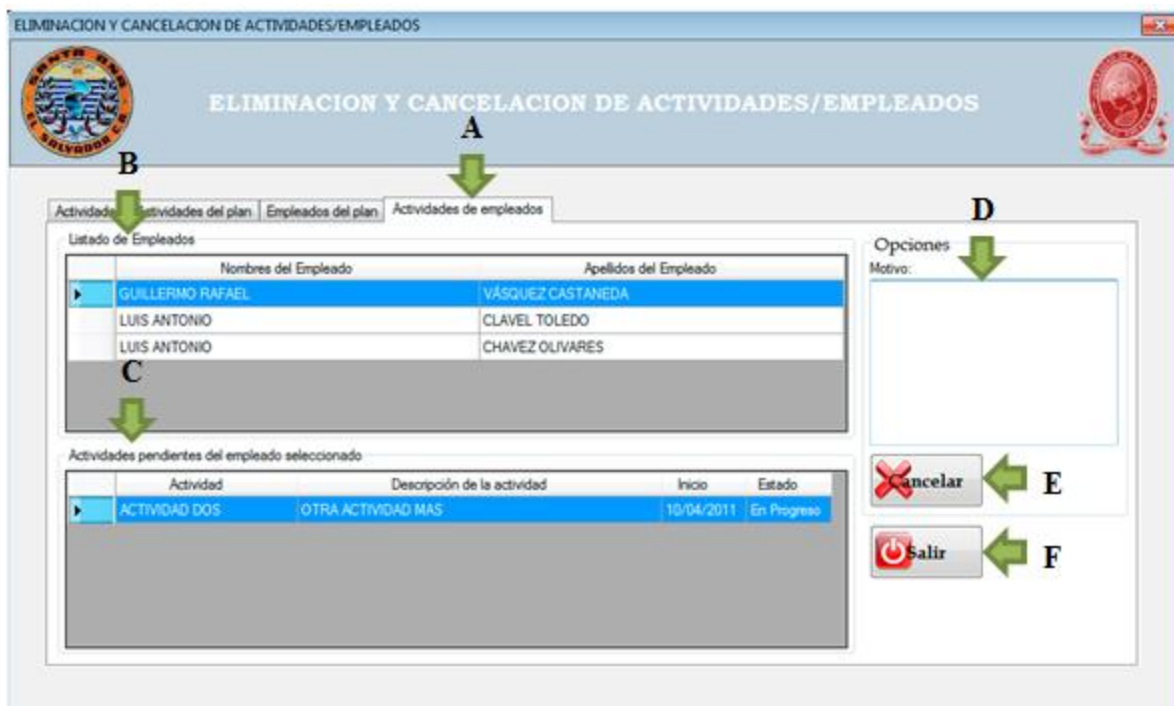


Figura 6.1.156 Cancelación de actividades de los empleados

- A.** Esta pestaña indica que se está trabajando con la cancelación de actividades de los empleados.
- B.** Muestra el listado de todos los empleados que contienen las actividades a cancelar. Debe seleccionar un empleado para poder acceder a sus actividades pendientes.
- C.** Muestra el listado de todas las actividades que pertenecen al empleado seleccionado y que están pendientes o han iniciado el día actual del sistema. Debe seleccionar la actividad que desea cancelar del empleado seleccionado.
- D.** En este campo debe de escribir el motivo por el cual cancela la actividad seleccionada, es obligatorio escribir el motivo de lo contrario el sistema impedirá que la actividad sea cancelada; cabe destacar que los caracteres especiales no están permitidos en este campo, además el número máximo es de 150 caracteres.
- E.** Este botón permite cancelar la actividad seleccionada (se puede hacer clic en este botón con la tecla Enter), siempre y cuando se haya ingresado un motivo para cancelar dicha actividad. Si se da el caso que no se ha escrito un motivo para la cancelación el sistema muestra el siguiente mensaje de error:

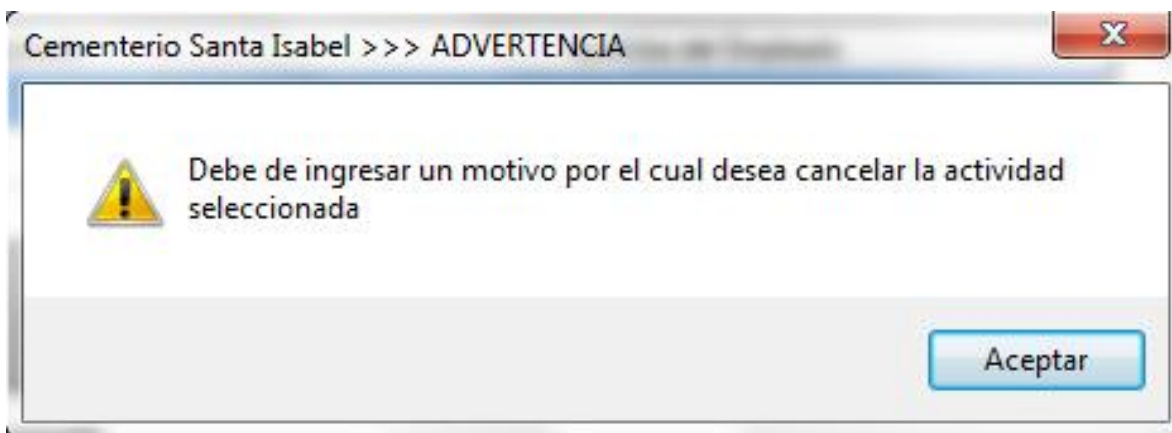


Figura 6.1.157 Mensaje de Error por no escribir un motivo

Si se ha escrito un motivo para la cancelación, el sistema nos preguntara si estamos seguros de cancelar la actividad por medio del siguiente mensaje:

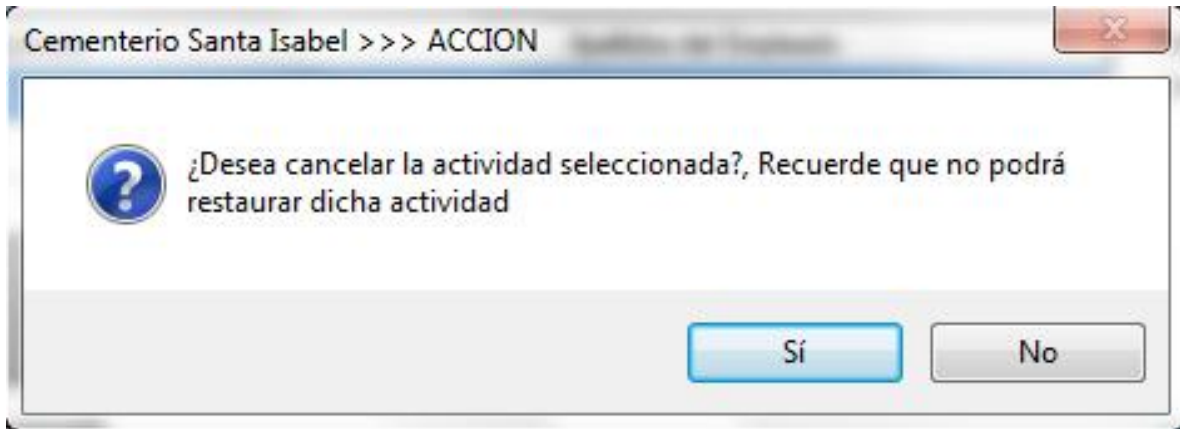


Figura 6.1.158 Mensaje preguntando por la cancelación de la actividad

En caso que respondamos afirmativamente el sistema nos mostrara el siguiente mensaje de confirmación:

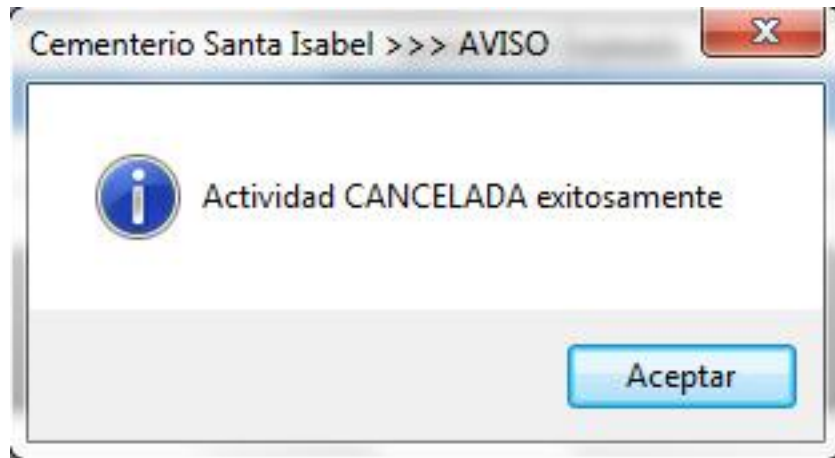


Figura 6.1.159 Mensaje confirmando la cancelación

F. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.156. En caso que queramos salir nos mostrara el mensaje correspondiente a la figura 6.1.98

Sub-opción Finalizar Actividades

Permite finalizar las actividades que se encuentran en progreso.

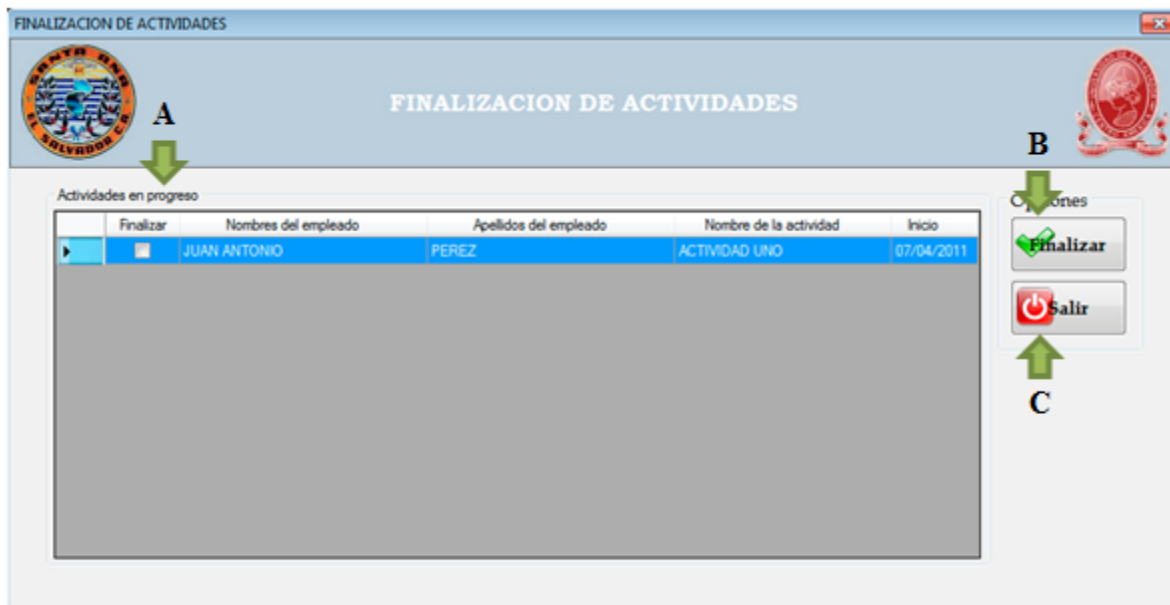


Figura 6.1.160 Finalización de actividades

A. Muestra el listado de todas las actividades que se encuentran en progreso; notara que al inicio de cada actividad hay una cajita, el objetivo acá es marcar las cajitas que correspondan a las actividades que se desea finalizar, lo que significa que deberá marcar todas las cajitas correspondientes a las actividades que vaya a finalizar.

B. Este botón permite finalizar las actividades que han sido marcadas, es decir las que tienen marcadas sus respectivas cajitas. El botón finalizar solamente estará habilitado si existe al menos una actividad en progreso, de lo contrario su estado será inhabilitado. Si se diera el caso que se hace clic en este botón sin marcar ninguna actividad, el sistema mostrara el siguiente mensaje de error:

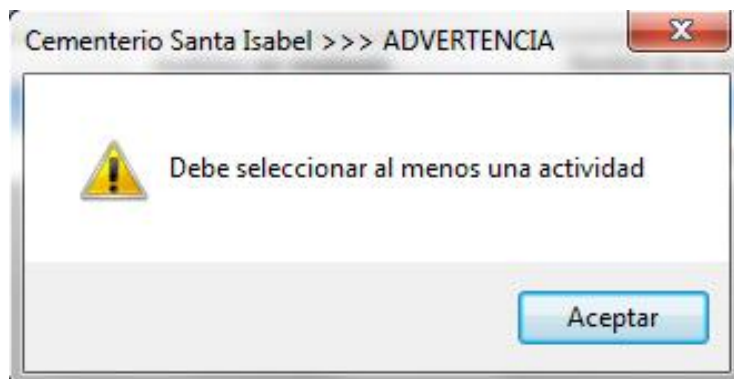


Figura 6.1.161 Mensaje de Error por no seleccionar ninguna actividad

Ahora bien, si se marcó al menos una actividad, se nos preguntará si estamos seguros de finalizar las actividades marcadas tal como lo muestra la siguiente imagen:

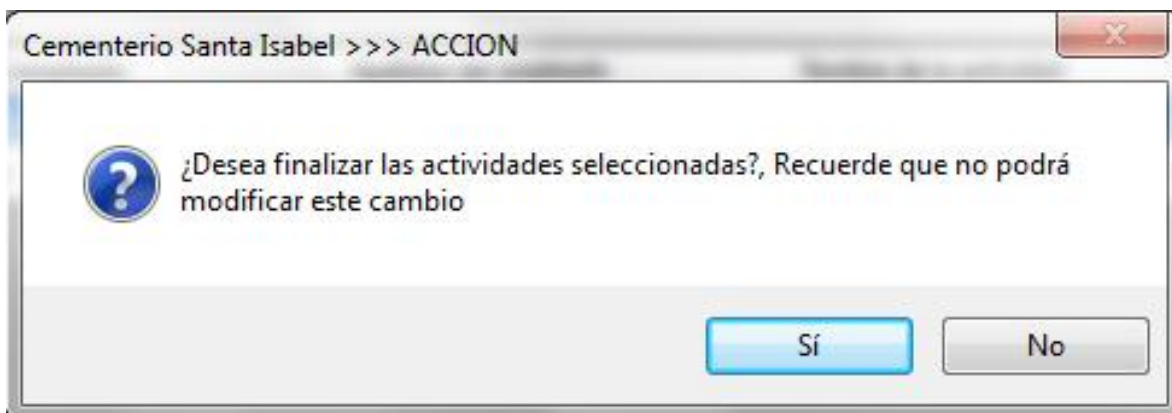


Figura 6.1.162 Mensaje preguntando por la finalización de actividades

Si respondimos afirmativamente la pregunta anterior, el sistema nos confirmará que las actividades se finalizaron tal como lo muestra la siguiente imagen:

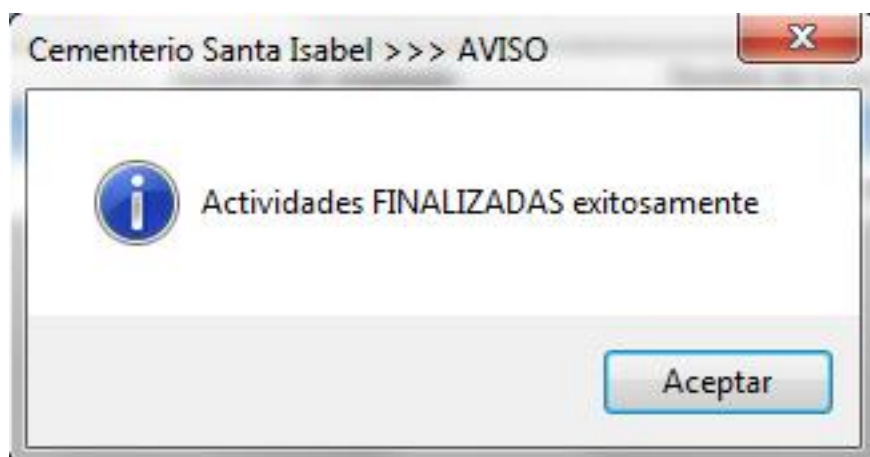


Figura 6.1.163 Mensaje confirmando finalización de actividades

C. Botón (se puede hacer clic en este botón con la tecla Esc) que nos permite salir de la pantalla que muestra la figura 6.1.160. En caso que queramos salir nos mostrará el mensaje correspondiente a la figura 6.1.98

MENÚ AYUDA

Permite obtener ayuda del sistema para tener una guía en caso de alguna duda, a la vez proporciona información general sobre el sistema.

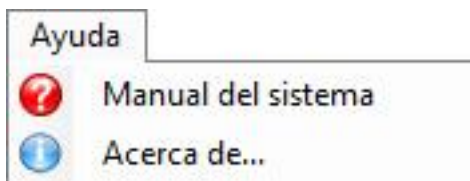


Figura 6.1.164 Ayuda

OPCIÓN MANUAL DEL SISTEMA

Muestra todo el manual de usuario dividido en secciones, es decir que se trata de este mismo manual de usuario que se encuentra leyendo en estos momentos, pero además esta ayuda brinda la posibilidad de poder ser impresa. Es importante mencionar que se puede acceder al manual directamente presionando la tecla F1.



Figura 6.1.165 Opción Manual del sistema

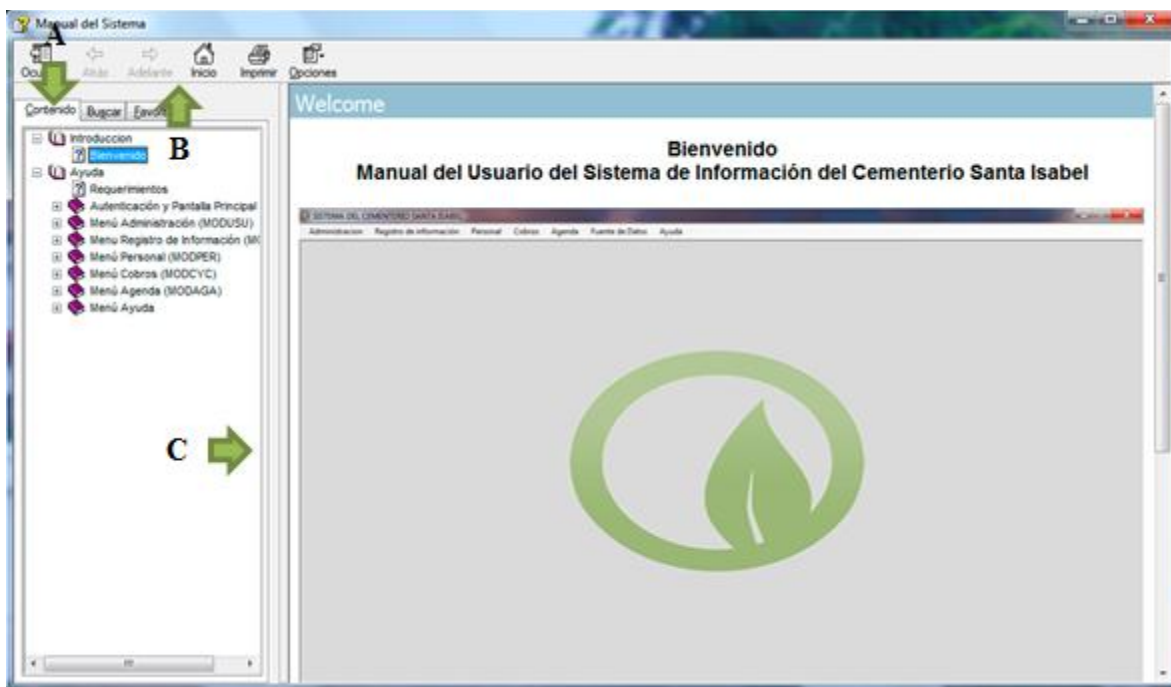


Figura 6.1.166 Manual de usuario

A. Estas pestañas indican las diferentes opciones para la navegación dentro del manual, estas se explican a continuación:

- 1) Contenido: Permite el acceso al contenido del manual en forma de árbol, es decir dividido por secciones, lo cual facilita la navegación por el manual.
- 2) Buscar: Permite buscar dentro del manual, para este propósito se proporcionan palabras que se desean buscar y si hay resultados se mostraran inmediatamente.
- 3) Favoritos: Nos permite agregar lugares favoritos, es decir lugares del manual que más frecuentamos para un fácil acceso a los mismos.

B. En esta barra se encuentran las opciones más comunes sobre el manual del sistema, estas se explican a continuación:

- 1) Ocultar: Permite ocultar las pestañas que nos brindan opciones sobre la navegación dentro del manual así como también hacerlas visibles en caso que estén ocultas.
- 2) Flechas: La que nos indica atrás permite retroceder una página y la que nos indica adelante permite adelantar una página.
- 3) Inicio: Nos permite visualizar la pantalla de inicio, es decir la bienvenida al manual del sistema.
- 4) Imprimir: Permite imprimir ya sea el manual completo o parte de este.
- 5) Opciones: Nos permite acceder a otras opciones que no están visibles por defecto.

C. Muestra el contenido del manual del sistema, el cual puede ser impreso.

OPCIÓN ACERCA DE...

Muestra información en general sobre el sistema, incluyendo sus desarrolladores, su versión, etc.

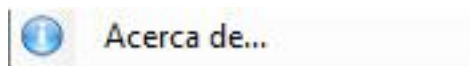


Figura 6.1.167 Opción Acerca de...



Figura 6.1.168 Acerca de...

6.2 MANUAL DE ADMINISTRACIÓN DEL SISTEMA DE INFORMACIÓN

CLASES DE LA CAPA DE APLICACIÓN

Acá se muestran todas las clases utilizadas por la capa de aplicación, la cual es la encargada de comunicarse con la base de datos y establece el flujo de información, a la vez que sirve de conexión entre la base de datos y la aplicación.

CIConexionCementerio: Clase utilizada para poder establecer una conexión con la base de datos, utilizada en la capa de aplicación, la cual es la encargada.

```
Imports System.IO
Imports System.Collections

Public Class CIConexionCementerio
#Region "Campos"
    Dim DmUsuario As String
    Dim DmServidor As String
    Dim DmDataBase As String
    Dim DmPassword As String
    Dim DmModulo As Integer
    Dim DmConectar As New SqlClient.SqlConnection
    Dim Dmestado As Boolean = False
#End Region

#Region "Propiedades"
    Public Property XUsuario() As String
        Get
            Return DmUsuario
        End Get
    End Property
End Class
```



```

        End Get
        Set(ByVal Value As String)
            'aquí puedo validar si el usuario digito numeros por ejemplo
            DmUsuario = Value
        End Set
    End Property

    Public Property XServidor() As String
        Get
            Return DmServidor
        End Get
        Set(ByVal Value As String)
            'aquí puedo validar si el usuario digito numeros por ejemplo
            DmServidor = Value
        End Set
    End Property

    Public Property XPassword() As String
        Get
            Return DmPassword
        End Get
        Set(ByVal Value As String)
            DmPassword = Value
        End Set
    End Property

    Public Property XDataBase() As String
        Get
            Return DmDataBase
        End Get
        Set(ByVal Value As String)
            DmDataBase = Value
        End Set
    End Property

    Public ReadOnly Property Estado() As Boolean
        Get
            Return Dmestado
        End Get
    End Property
#End Region

#Region "Métodos"
    Public Function ConectoSQL() As SqlConnection

        Dim PathArchivo As String = Application.StartupPath + "\ValCon.dat"

        If File.Exists(PathArchivo) Then
            Dim objReader As New StreamReader(PathArchivo)
            Dim sLine As String = ""
            Dim arrText As New ArrayList()

            Do
                sLine = objReader.ReadLine()
                If Not sLine Is Nothing Then
                    arrText.Add(sLine)
                End If
            Loop
        End If
    End Function

```

```

Loop Until sLine Is Nothing
objReader.Close()

DmServidor = arrText.Item(0).ToString
'DmServidor = My.Computer.Name + "\SRVCEMENTERIO" 'este parametro es el
nombre del servidor, el cual debe ser estandar podremos a la instancia el nombre
SRVCEMENTERIO,asi que uds deben nombrar asi a la instancia del SQL
DmDataBase = "DBCEMENTERIO" 'El nombre de la base de datos que para el
caso sera DBCEMENTERIO
DmConectar.ConnectionString = "integrated security = true; Data source
=" & DmServidor & ";persist security info = False; initial catalog =" & DmDataBase
'DmConectar.ConnectionString = "Data Source=" & DmServidor & ";Initial
Catalog=" & DmDataBase & ";User ID=cementerio;Password=cementerio"
End If

Try
DmConectar.Open()
Dmestado = True
Return DmConectar
Catch ex As Exception
Dmestado = False
DmConectar.Close()
MsgBox("El intento de conexion no fue exitoso")
Return DmConectar
End Try
End Function

Public Sub CerrarConexion()
DmConectar.Close()
End Sub
#End Region
End Class

```

CLASES DEL MODULO DE USUARIOS (MODUSU)

CIUsuarios: Clase utilizada para poder establecer una conexión con la tablas de usuarios y roles del sistema.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Imports System.Security.Cryptography
Public Class CIUsuarios
#Region "Campos"
Protected DmCodigoUsuario As Integer
Protected DmUsuario As String
Protected DmPassword As String
Protected DmEstado As Boolean
Protected DmRol As Integer
Protected DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
Public Property CodigoUsuario() As Integer 'Codigo de usuario
Get
Return DmCodigoUsuario

```

```

        End Get
        Set(ByVal Value As Integer)
            DmCodigoUsuario = Value
        End Set
    End Property

    Public Property Usuario() As String ' Nombre de usuario
        Get
            Return DmUsuario
        End Get
        Set(ByVal Value As String)
            DmUsuario = Value
        End Set
    End Property

    Public Property Password() As String ' Password de usuario
        Get
            Return DmPassword
        End Get
        Set(ByVal Value As String)
            DmPassword = Value
        End Set
    End Property

    Public Property Estado() As Boolean ' Estado del usuario
        Get
            Return DmEstado
        End Get
        Set(ByVal Value As Boolean)
            DmEstado = Value
        End Set
    End Property

    Public Property Rol() As Integer 'Codigo del rol del usuario
        Get
            Return DmRol
        End Get
        Set(ByVal Value As Integer)
            DmRol = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)
            DmSqlCon = Value
        End Set
    End Property
#End Region

#Region "Métodos"
    Public Function MostrarUsuario() As Boolean ' Comprueba si el usuario puede
    ingresar al sistema o no
        Dim cmd As New SqlCommand("MODUSU.SP_VERUSUARIO", DmSqlCon)
        Dim data As SqlDataReader

```

```

    Try
        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@CodigoUsuario", SqlDbType.Int).Value =
DmCodigoUsuario
        data = cmd.ExecuteReader()
        If data.Read Then
            If Not IsDBNull(data.Item("CodUsuario")) Then DmCodigoUsuario =
data.Item("CodUsuario")
            If Not IsDBNull(data.Item("Usuario")) Then DmUsuario =
data.Item("Usuario")
            If Not IsDBNull(data.Item("CodRol")) Then DmRol =
data.Item("CodRol")
            If Not IsDBNull(data.Item("Estado")) Then DmEstado =
data.Item("Estado")
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

```

```

    Public Function MostrarEmpleados() As DataSet ' Muestra los nombres y
apellidos de los empleados en base a su estado
    Dim adp As New SqlDataAdapter("MODPER.SP_VEREMPLEADOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

```

```

    Public Function MostrarRoles() As DataSet ' Muestra los roles en base a su
estado
    Dim adp As New SqlDataAdapter("MODUSU.SP_VERROLES", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
    End Try

```

```

        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function GuardarUsuario() As Boolean ' Agrega un nuevo usuario, tanto
a la tabla de usuario, asi como tmb se relaciona con el empleado
    Dim cmd As New SqlCommand("MODUSU.SP_GUARDARUSUARIO", DmSqlCon)
    cmd.Parameters.Add("@Empleado", SqlDbType.Int).Value = DmCodigoUsuario
    cmd.Parameters.Add("@Usuario", SqlDbType.VarChar, 20).Value = DmUsuario
    cmd.Parameters.Add("@Password", SqlDbType.VarChar, 128).Value =
StrToHash(DmPassword)
    cmd.Parameters.Add("@Rol", SqlDbType.Int).Value = DmRol
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarUsuario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ModificarPasswordUsuario() As Boolean ' Modifica el password del
usuario
    Dim cmd As New SqlCommand("MODUSU.SP_MODIFICARPASSWORD", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoUsuario
    cmd.Parameters.Add("@Password", SqlDbType.VarChar, 128).Value =
StrToHash(DmPassword)
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarPasswordUsuario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ModificarUsuario() As Boolean ' Modifica el usuario seleccionado
    Dim cmd As New SqlCommand("MODUSU.SP_MODIFICARUSUARIO", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoUsuario
    cmd.Parameters.Add("@Rol", SqlDbType.Int).Value = DmRol
    cmd.Parameters.Add("@Estado", SqlDbType.Bit).Value = DmEstado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarUsuario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

```

```

Public Function VerificarUsuario() As Boolean ' Verifica si un usuario esta
siendo ingresado dos veces
Dim cmd As New SqlCommand("MODUSU.SP_VERIFICARUSUARIO", DmSqlCon)
Dim data As SqlDataReader
Try
cmd.CommandType = CommandType.StoredProcedure
cmd.Parameters.Add("@Usuario", SqlDbType.VarChar, 20).Value = DmUsuario
data = cmd.ExecuteReader()
If data.Read Then
Return True
Else
Return False
End If
Catch
Throw
Finally
cmd.Dispose()
DmSqlCon.Close()
End Try
End Function

Public Function UsuarioCorrecto() As Boolean ' Comprueba si el usuario puede
ingresar al sistema o no
Dim cmd As New SqlCommand("MODUSU.SP_USUARIOCORRECTO", DmSqlCon)
Dim data As SqlDataReader
Dim UsrFrm As String = Nothing
Dim PassFrm As String = Nothing
Try
cmd.CommandType = CommandType.StoredProcedure
UsrFrm = DmUsuario
PassFrm = StrToHash(DmPassword)
cmd.Parameters.Add("@Usuario", SqlDbType.VarChar, 20).Value = UsrFrm
cmd.Parameters.Add("@Password", SqlDbType.VarChar, 128).Value = PassFrm
data = cmd.ExecuteReader()
If data.Read Then
If Not IsDBNull(data.Item("CodUsuario")) Then DmCodigoUsuario =
data.Item("CodUsuario")
If Not IsDBNull(data.Item("Usuario")) Then DmUsuario =
data.Item("Usuario")
If Not IsDBNull(data.Item("Password")) Then DmPassword =
data.Item("Password")
If Not IsDBNull(data.Item("Estado")) Then DmEstado =
data.Item("Estado")
If Not IsDBNull(data.Item("Rol")) Then DmRol = data.Item("Rol")
End If
If (UsrFrm = DmUsuario) And (PassFrm = DmPassword) Then
Return True
Else
Return False
End If
Catch
Throw
Finally
cmd.Dispose()
DmSqlCon.Dispose()
End Try
End Function

```

```

    Public Function StrToHash(ByVal TextToHash As String) As String ' Convierte
de String a Hash cualquier cadena de texto
    Dim bytValue As Byte()
    Dim bytHash As Byte()
    Dim mHash As MD5CryptoServiceProvider
    mHash = New MD5CryptoServiceProvider
    bytValue = System.Text.Encoding.UTF8.GetBytes(TextToHash)
    bytHash = mHash.ComputeHash(bytValue)
    mHash.Clear()
    Return Convert.ToBase64String(bytHash)
End Function
#End Region
End Class

```

CLASES DEL MODULO DE REGISTRO DE INFORMACIÓN (MODRIN)

ClBeneficiarios: Clase encargada de manipular los beneficiarios de una propiedad.

```

Imports System.Data.SqlClient
Public Class ClBeneficiarios
#Region "Campos"
    Private DmCodBeneficiario As Integer
    Private DmCodPropiedad As Integer
    Private DmNombres As String
    Private DmApellidos As String
    Private DmNumeroDui As String
    Private DmEstado As Boolean
    Private DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
    Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal value As SqlConnection)
        DmSqlCon = value
    End Set
End Property

    Public Property CodBeneficiario() As Integer
    Get
        Return DmCodBeneficiario
    End Get
    Set(ByVal Value As Integer)
        DmCodBeneficiario = Value
    End Set
End Property

    Public Property CodPropiedad() As Integer
    Get
        Return DmCodPropiedad
    End Get
    Set(ByVal Value As Integer)
        DmCodPropiedad = Value
    End Set
End Property

```

```

        End Set
    End Property

    Public Property Nombres() As String
        Get
            Return DmNombres
        End Get
        Set(ByVal Value As String)
            DmNombres = Value
        End Set
    End Property

    Public Property Apellidos() As String
        Get
            Return DmApellidos
        End Get
        Set(ByVal Value As String)
            DmApellidos = Value
        End Set
    End Property

    Public Property NumeroDui() As String
        Get
            Return DmNumeroDui
        End Get
        Set(ByVal Value As String)
            DmNumeroDui = Value
        End Set
    End Property

    Public Property Estado() As Boolean
        Get
            Return DmEstado
        End Get
        Set(ByVal Value As Boolean)
            DmEstado = Value
        End Set
    End Property
#End Region

#Region "Métodos"
    Public Function ListarBeneficiarios() As DataSet ' Muestra los beneficiarios
    por propiedad
        Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARBENEFICIARIOS", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Propiedad", SqlDbType.Int).Value =
DmCodPropiedad
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
        End Try
        Return ds
    End Function

```



```

Public Sub BuscarBeneficiario()
    Dim cmd As New SqlCommand("SP_BUSCARBENEFICIARIO", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then
            If Not IsDBNull(data.Item("CodBeneficiario")) Then DmCodBeneficiario
= data.Item("CodBeneficiario")
            If Not IsDBNull(data.Item("CodPropiedad")) Then DmCodPropiedad =
data.Item("CodPropiedad")
            If Not IsDBNull(data.Item("Nombres")) Then DmNombres =
data.Item("Nombres")
            If Not IsDBNull(data.Item("Apellidos")) Then DmApellidos =
data.Item("Apellidos")
            If Not IsDBNull(data.Item("NumeroDui")) Then DmNumeroDui =
data.Item("NumeroDui")
            End If
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Sub

Public Function GuardarBeneficiario() As Boolean ' Guarda un nuevo
beneficiario
    Dim cmd As New SqlCommand("MODRIN.SP_GUARDARBENEFICIARIO", DmSqlCon)
    cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value = DmCodPropiedad
    cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value = DmNumeroDui
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarBeneficiario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Function

Public Function ModificarBeneficiario() As Boolean ' Modifica el beneficiario
seleccionado
    Dim cmd As New SqlCommand("MODRIN.SP_MODIFICARBENEFICIARIO", DmSqlCon)
    cmd.Parameters.Add("@CodBeneficiario", SqlDbType.Int).Value =
DmCodBeneficiario
    cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value = DmCodPropiedad
    cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value = DmNumeroDui
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarBeneficiario = cmd.ExecuteNonQuery()
    Catch

```

```

        Throw
    Finally
        cmd.Dispose()
    End Try
End Function

Public Function EliminarBeneficiario() As Boolean
    Dim cmd As New SqlCommand("SP_ELIMINARBENEFICIARIO", DmSqlCon)
    Try
        cmd.CommandType = CommandType.StoredProcedure
        EliminarBeneficiario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Function

Public Function VerificarBeneficiario() As Boolean ' Verifica si un mismo
beneficiario esta siendo ingresado
    Dim cmd As New SqlCommand("MODRIN.SP_VERIFICARBENEFICIARIO", DmSqlCon) '
dos veces en una misma propiedad
    Dim data As SqlDataReader
    Try
        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value =
DmCodPropiedad
        cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value =
DmNumeroDui
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Close()
    End Try
End Function
#End Region
End Class

```

CIClientes: se encarga de gestionar todas las operaciones que se realizan con los clientes del Cementerio Santa Isabel.

```
Imports System.Data.SqlClient
```

```
Public Class CIClientes
```

```

    Private DmCodCliente As Integer
    Private DmNombres As String
    Private DmApellidos As String
    Private DmEstadoCivil As String

```

```

Private DmDireccion As String
Private DmNumeroDui As String
Private DmLugarExtensionDui As String
Private DmFechaNacimiento As DateTime
Private DmTelefono As String
Private DmDptoExtDui As String
Private DmSqlCon As SqlConnection

Public Property GetConexion As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal value As SqlConnection)
        DmSqlCon = value
    End Set
End Property

Public Property CodCliente() As Integer
    Get
        Return DmCodCliente
    End Get
    Set(ByVal Value As Integer)
        DmCodCliente = Value
    End Set
End Property

Public Property Nombres() As String
    Get
        Return DmNombres
    End Get
    Set(ByVal Value As String)
        DmNombres = Value
    End Set
End Property

Public Property Apellidos() As String
    Get
        Return DmApellidos
    End Get
    Set(ByVal Value As String)
        DmApellidos = Value
    End Set
End Property

Public Property EstadoCivil() As String
    Get
        Return DmEstadoCivil
    End Get
    Set(ByVal Value As String)
        DmEstadoCivil = Value
    End Set
End Property

Public Property Direccion() As String
    Get
        Return DmDireccion
    End Get

```

```

        Set(ByVal Value As String)
            DmDireccion = Value
        End Set
    End Property

    Public Property NumeroDui() As String
        Get
            Return DmNumeroDui
        End Get
        Set(ByVal Value As String)
            DmNumeroDui = Value
        End Set
    End Property

    Public Property LugarExtensionDui() As String
        Get
            Return DmLugarExtensionDui
        End Get
        Set(ByVal Value As String)
            DmLugarExtensionDui = Value
        End Set
    End Property

    Public Property FechaNacimiento() As DateTime
        Get
            Return DmFechaNacimiento
        End Get
        Set(ByVal Value As DateTime)
            DmFechaNacimiento = Value
        End Set
    End Property

    Public Property Telefono() As String
        Get
            Return DmTelefono
        End Get
        Set(ByVal Value As String)
            DmTelefono = Value
        End Set
    End Property

    Public Property DptoExtDui As String
        Get
            Return DmDptoExtDui
        End Get
        Set(ByVal value As String)
            DmDptoExtDui = value
        End Set
    End Property

    Public Function VerificarCliente() As Boolean

        Dim cmd As New SqlCommand("MODRIN.SP_VERIFICARCLIENTE", DmSqlCon)
        Dim data As SqlDataReader

        Try

```

```

        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value =
DmNumeroDui
        data = cmd.ExecuteReader()

        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Close()
    End Try
End Function

```

```

Public Function ListarClientes() As DataSet

```

```

    Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARCLIENTES", DmSqlCon)
    Dim ds As New DataSet

```

```

    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try

```

```

    Return ds
End Function

```

```

Public Sub BuscarCliente()

```

```

    Dim cmd As New SqlCommand("MODRIN.SP_BUSCARCLIENTE", DmSqlCon)
    Dim data As SqlDataReader

```

```

    cmd.CommandType = CommandType.StoredProcedure
    data = cmd.ExecuteReader()

```

```

    If data.Read Then
        If Not IsDBNull(data.Item("CodCliente")) Then DmCodCliente =
data.Item("CodCliente")
        If Not IsDBNull(data.Item("Nombres")) Then DmNombres =
data.Item("Nombres")
        If Not IsDBNull(data.Item("Apellidos")) Then DmApellidos =
data.Item("Apellidos")
        If Not IsDBNull(data.Item("EstadoCivil")) Then DmEstadoCivil =
data.Item("EstadoCivil")
        If Not IsDBNull(data.Item("Direccion")) Then DmDireccion =
data.Item("Direccion")
    End If

```

```

        If Not IsDBNull(data.Item("NumeroDui")) Then DmNumeroDui =
data.Item("NumeroDui")
        If Not IsDBNull(data.Item("LugarExtensionDui")) Then DmLugarExtensionDui
= data.Item("LugarExtensionDui")
        If Not IsDBNull(data.Item("FechaNacimiento")) Then DmFechaNacimiento =
data.Item("FechaNacimiento")
        If Not IsDBNull(data.Item("DptoExtDui")) Then DmDptoExtDui =
data.Item("DptoExtDui")
        If Not IsDBNull(data.Item("Telefono")) Then DmTelefono =
data.Item("Telefono")
    End If

```

```
End Sub
```

```
Public Function GuardarCliente() As Boolean
```

```

    Dim cmd As New SqlCommand("MODRIN.SP_GUARDARCLIENTE", DmSqlCon)

    cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@EstadoCivil", SqlDbType.VarChar, 1).Value =
DmEstadoCivil
    cmd.Parameters.Add("@Direccion", SqlDbType.VarChar, 350).Value = DmDireccion
    cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value = DmNumeroDui
    cmd.Parameters.Add("@LugarExtensionDui", SqlDbType.VarChar, 50).Value =
DmLugarExtensionDui
    cmd.Parameters.Add("@FechaNacimiento", SqlDbType.DateTime).Value =
DmFechaNacimiento
    cmd.Parameters.Add("@Telefono", SqlDbType.VarChar, 9).Value = DmTelefono
    cmd.Parameters.Add("@DptoExtDui", SqlDbType.VarChar, 150).Value =
DmDptoExtDui
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarCliente = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try

```

```
End Function
```

```
Public Function ModificarCliente() As Boolean
```

```

    Dim cmd As New SqlCommand("MODRIN.SP_MODIFICARCLIENTES", DmSqlCon)

    cmd.Parameters.Add("@CodCliente", SqlDbType.Int).Value = DmCodCliente
    cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@EstadoCivil", SqlDbType.VarChar, 1).Value =
DmEstadoCivil
    cmd.Parameters.Add("@Direccion", SqlDbType.VarChar, 350).Value = DmDireccion
    cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 50).Value = DmNumeroDui
    cmd.Parameters.Add("@LugarExtensionDui", SqlDbType.VarChar, 50).Value =
DmLugarExtensionDui

```

```

        cmd.Parameters.Add("@FechaNacimiento", SqlDbType.DateTime).Value =
DmFechaNacimiento
        cmd.Parameters.Add("@Telefono", SqlDbType.VarChar, 9).Value = DmTelefono
        cmd.Parameters.Add("@DptoExtDui", SqlDbType.VarChar, 150).Value =
DmDptoExtDui
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarCliente = cmd.ExecuteNonQuery()

    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try

End Function

Public Function EliminarCliente() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_ELIMINARCLIENTE", DmSqlCon)

    Try
        cmd.CommandType = CommandType.StoredProcedure
        EliminarCliente = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

End Class

```

CIExhumacion: Clase encargada de gestionar todas las transacciones relacionadas a las exhumaciones que registra el Cementerio Santa Isabel.

```

Imports System.Data.SqlClient

Public Class CIExhumacion
#Region "Variables"
    Private DmCodExhumacion As Integer
    Private DmCodInhumacion As Integer
    Private DmFechaInhumacion As DateTime
    Private DmObservaciones As String
    Private DmCausaInhumacion As String
    Private DmCodEmpleado As Integer
    Private DmSqlConec As SqlConnection
#End Region

#Region "Property's"
    Public Property CodExhumacion() As Integer
    Get
        Return DmCodExhumacion
    End Get
End Property

```

```

        End Get
        Set(ByVal Value As Integer)
            DmCodExhumacion = Value
        End Set
    End Property

    Public Property CodInhumacion() As Integer
        Get
            Return DmCodInhumacion
        End Get
        Set(ByVal Value As Integer)
            DmCodInhumacion = Value
        End Set
    End Property

    Public Property FechaInhumacion() As DateTime
        Get
            Return DmFechaInhumacion
        End Get
        Set(ByVal Value As DateTime)
            DmFechaInhumacion = Value
        End Set
    End Property

    Public Property Observaciones() As String
        Get
            Return DmObservaciones
        End Get
        Set(ByVal Value As String)
            DmObservaciones = Value
        End Set
    End Property

    Public Property CausaInhumacion() As String
        Get
            Return DmCausaInhumacion
        End Get
        Set(ByVal Value As String)
            DmCausaInhumacion = Value
        End Set
    End Property

    Public Property CodEmpleado() As Integer
        Get
            Return DmCodEmpleado
        End Get
        Set(ByVal Value As Integer)
            DmCodEmpleado = Value
        End Set
    End Property

    Public Property GetConexion As SqlConnection
        Get
            Return DmSqlConec
        End Get
        Set(ByVal value As SqlConnection)
            DmSqlConec = value
        End Set
    End Property

```



```

        End Set
    End Property
#End Region

#Region "Funciones"
    Public Sub BuscarInhumacionCodigo()
        Dim cmd As New SqlCommand("SP_BUSCARINHUMACIONCODIGO", DmSqlConec)
        Dim data As SqlDataReader
        Try
            cmd.CommandType = CommandType.StoredProcedure
            data = cmd.ExecuteReader()
            If data.Read Then
                If Not IsDBNull(data.Item("CodExhumacion")) Then DmCodExhumacion =
data.Item("CodExhumacion")
                If Not IsDBNull(data.Item("CodInhumacion")) Then DmCodInhumacion =
data.Item("CodInhumacion")
                If Not IsDBNull(data.Item("FechaInhumacion")) Then DmFechaInhumacion
= data.Item("FechaInhumacion")
                If Not IsDBNull(data.Item("Observaciones")) Then DmObservaciones =
data.Item("Observaciones")
                If Not IsDBNull(data.Item("CausaInhumacion")) Then DmCausaInhumacion
= data.Item("CausaInhumacion")
                If Not IsDBNull(data.Item("CodEmpleado")) Then DmCodEmpleado =
data.Item("CodEmpleado")
            End If
        Catch
            Throw
        Finally
            cmd.Dispose()
        End Try
    End Sub

    Public Function GuardarExhumacion() As Boolean ' Guarda una nueva exhumación
        Dim cmd As New SqlCommand("MODRIN.SP_GUARDAREXHUMACION", DmSqlConec)
        cmd.Parameters.Add("@CodInhumacion", SqlDbType.Int).Value = DmCodInhumacion
        cmd.Parameters.Add("@FechaInhumacion", SqlDbType.DateTime).Value =
DmFechaInhumacion
        cmd.Parameters.Add("@Observaciones", SqlDbType.VarChar, 350).Value =
DmObservaciones
        cmd.Parameters.Add("@CausaInhumacion", SqlDbType.VarChar, 50).Value =
DmCausaInhumacion
        cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
        Try
            cmd.CommandType = CommandType.StoredProcedure
            GuardarExhumacion = cmd.ExecuteNonQuery()
        Catch
            Throw
        Finally
            cmd.Dispose()
        End Try
    End Function

    Public Function ModificarExhumacion() As Boolean
        Dim cmd As New SqlCommand("Dm_TBLEXHUMACIONES_Modificar", DmSqlConec)
        cmd.Parameters.Add("@CodExhumacion", SqlDbType.Int).Value = DmCodExhumacion
        cmd.Parameters.Add("@CodInhumacion", SqlDbType.Int).Value = DmCodInhumacion

```

```

        cmd.Parameters.Add("@FechaInhumacion", SqlDbType.DateTime).Value =
DmFechaInhumacion
        cmd.Parameters.Add("@Observaciones", SqlDbType.VarChar, 350).Value =
DmObservaciones
        cmd.Parameters.Add("@CausaInhumacion", SqlDbType.VarChar, 50).Value =
DmCausaInhumacion
        cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarExhumacion = cmd.ExecuteNonQuery
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Function
#End Region
End Class

```

CIInhumacion: Clase encargada de gestionar todas las transacciones relacionadas con el registro de las inhumaciones que procesa el Cementerio Santa Isabel.

```

Imports System.Data.SqlClient
Public Class CIInhumacion
#Region "Variables"
    Private DmCodInhumacion As Integer
    Private DmCodIsam As Integer
    Private DmNumeroPartidaDefuncion As String
    Private DmCodPropiedad As Integer
    Private DmCodEmpleado As Integer
    Private DmCausasFallecimiento As String
    Private DmLugarFallecimiento As String
    Private DmDireccionFallecimiento As String
    Private DmFechaInhumacion As DateTime
    Private DmSqlCon As SqlConnection
#End Region

#Region "Property's"
    Public Property CodInhumacion() As Integer
    Get
        Return DmCodInhumacion
    End Get
    Set(ByVal Value As Integer)
        DmCodInhumacion = Value
    End Set
End Property

    Public Property CodIsam() As Integer
    Get
        Return DmCodIsam
    End Get
    Set(ByVal Value As Integer)
        DmCodIsam = Value
    End Set

```

```

End Property

Public Property NumeroPartidaDefuncion() As String
    Get
        Return DmNumeroPartidaDefuncion
    End Get
    Set(ByVal Value As String)
        DmNumeroPartidaDefuncion = Value
    End Set
End Property

Public Property CodPropiedad() As Integer
    Get
        Return DmCodPropiedad
    End Get
    Set(ByVal Value As Integer)
        DmCodPropiedad = Value
    End Set
End Property

Public Property CodEmpleado() As Integer
    Get
        Return DmCodEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodEmpleado = Value
    End Set
End Property

Public Property CausasFallecimiento() As String
    Get
        Return DmCausasFallecimiento
    End Get
    Set(ByVal Value As String)
        DmCausasFallecimiento = Value
    End Set
End Property

Public Property LugarFallecimiento() As String
    Get
        Return DmLugarFallecimiento
    End Get
    Set(ByVal Value As String)
        DmLugarFallecimiento = Value
    End Set
End Property

Public Property DireccionFallecimiento() As String
    Get
        Return DmDireccionFallecimiento
    End Get
    Set(ByVal Value As String)
        DmDireccionFallecimiento = Value
    End Set
End Property

Public Property FechaInhumacion() As DateTime

```

```

    Get
        Return DmFechaInhumacion
    End Get
    Set(ByVal Value As DateTime)
        DmFechaInhumacion = Value
    End Set
End Property

Public Property GetConexion As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal value As SqlConnection)
        DmSqlCon = value
    End Set
End Property
#End Region

#Region "Funciones"

Public Function ObtenerDatosFallecido() As ClPartidaDefuncion
    Dim cmd As New SqlCommand("MODRIN.SP_OBTENERNOMBREFALLECIDO", DmSqlCon)
    Dim data As SqlDataReader
    Dim partida As New ClPartidaDefuncion

    cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.Int).Value =
DmNumeroPartidaDefuncion
    Try
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then
            If Not IsDBNull(data.Item("NombreFallecido")) Then
partida.NombreFallecido = data.Item("NombreFallecido")
            If Not IsDBNull(data.Item("OrigenFallecido")) Then
partida.OrigenFallecido = data.Item("OrigenFallecido")
            If Not IsDBNull(data.Item("DireccionFallecido")) Then
partida.DireccionFallecido = data.Item("DireccionFallecido")
            If Not IsDBNull(data.Item("NombrePadre")) Then partida.NombrePadre =
data.Item("NombrePadre")
            If Not IsDBNull(data.Item("NombreMadre")) Then partida.NombreMadre =
data.Item("NombreMadre")
            If Not IsDBNull(data.Item("MedicoAvalaDefuncion")) Then
partida.MedicoAvalaDefuncion = data.Item("MedicoAvalaDefuncion")
            If Not IsDBNull(data.Item("HoraFechaFallecimiento")) Then
partida.HoraFechaFallecimiento = data.Item("HoraFechaFallecimiento")
            If Not IsDBNull(data.Item("Genero")) Then partida.Genero =
data.Item("Genero")
            End If
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
    Return partida
End Function

Public Function ListarInhumaciones() As DataSet ' Lista todas las inhumaciones

```

```

Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARINHUMACIONES", DmSqlCon)
Dim ds As New DataSet
Try
    adp.SelectCommand.CommandType = CommandType.StoredProcedure
    adp.Fill(ds)
Catch
    Throw
Finally
    adp.Dispose()
End Try
Return ds
End Function

Public Sub BuscarInhumacionCodigo()
Dim cmd As New SqlCommand("SP_BUSCARINHUMACIONCODIGO", DmSqlCon)
Dim data As SqlDataReader
cmd.Parameters.Add("@CodInhumacion", SqlDbType.Int).Value = DmCodInhumacion
Try
    cmd.CommandType = CommandType.StoredProcedure
    data = cmd.ExecuteReader()
    If data.Read Then
        If Not IsDBNull(data.Item("CodIsam")) Then DmCodIsam =
data.Item("CodIsam")
        If Not IsDBNull(data.Item("NumeroPartidaDefuncion")) Then
DmNumeroPartidaDefuncion = data.Item("NumeroPartidaDefuncion")
        If Not IsDBNull(data.Item("CodPropiedad")) Then DmCodPropiedad =
data.Item("CodPropiedad")
        If Not IsDBNull(data.Item("CodEmpleado")) Then DmCodEmpleado =
data.Item("CodEmpleado")
        If Not IsDBNull(data.Item("CausasFallecimiento")) Then
DmCausasFallecimiento = data.Item("CausasFallecimiento")
        If Not IsDBNull(data.Item("LugarFallecimiento")) Then
DmLugarFallecimiento = data.Item("LugarFallecimiento")
        If Not IsDBNull(data.Item("DireccionFallecimiento")) Then
DmDireccionFallecimiento = data.Item("DireccionFallecimiento")
        If Not IsDBNull(data.Item("FechaInhumacion")) Then DmFechaInhumacion
= data.Item("FechaInhumacion")
    End If
Catch
    Throw
Finally
    cmd.Dispose()
End Try
End Sub

Public Function GuardarInhumacion() As Boolean ' Guarda una nueva inhumación
Dim cmd As New SqlCommand("MODRIN.SP_GUARDARINHUMACION", DmSqlCon)
cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value = DmCodPropiedad
cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
cmd.Parameters.Add("@CausasFallecimiento", SqlDbType.VarChar, 500).Value =
DmCausasFallecimiento
cmd.Parameters.Add("@LugarFallecimiento", SqlDbType.VarChar, 50).Value =
DmLugarFallecimiento

```

```

        cmd.Parameters.Add("@DireccionFallecimiento", SqlDbType.VarChar, 250).Value
= DmDireccionFallecimiento
        cmd.Parameters.Add("@FechaInhumacion", SqlDbType.DateTime).Value =
DmFechaInhumacion
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarInhumacion = cmd.ExecuteNonQuery
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Function

Public Function ModificarInhumacion() As Boolean
    Dim cmd As New SqlCommand("SP_MODIFICARINHUMACION", DmSqlCon)
    cmd.Parameters.Add("@CodInhumacion", SqlDbType.Int).Value = DmCodInhumacion
    cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
    cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
    cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value = DmCodPropiedad
    cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
    cmd.Parameters.Add("@CausasFallecimiento", SqlDbType.VarChar, 500).Value =
DmCausasFallecimiento
    cmd.Parameters.Add("@LugarFallecimiento", SqlDbType.VarChar, 50).Value =
DmLugarFallecimiento
    cmd.Parameters.Add("@DireccionFallecimiento", SqlDbType.VarChar, 250).Value
= DmDireccionFallecimiento
    cmd.Parameters.Add("@FechaInhumacion", SqlDbType.DateTime).Value =
DmFechaInhumacion
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarInhumacion = cmd.ExecuteNonQuery
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try
End Function
#End Region
End Class

```

CIPartidaDefuncion: Clase encargada de administrar todo lo relacionado a las partidas de defunción que son registradas y emitidas por el Cementerio Santa Isabel.

```

Imports System.Data.SqlClient
Public Class CIPartidaDefuncion
#Region "Variables"
    Private DmNumeroPartidaDefuncion As String
    Private DmCodIsam As Integer
    Private DmNombreSolicitante As String
    Private DmNombreFallecido As String
    Private DmEdad As Integer
    Private DmEstadoCivil As String
    Private DmOrigenFallecido As String

```

```

Private DmDireccionFallecido As String
Private DmOficioProfesionFallecidos As String
Private DmNombrePadre As String
Private DmNombreMadre As String
Private DmMedicoAvalaDefuncion As String
Private DmHoraFechaFallecimiento As DateTime
Private DmCodEmpleado As Integer
Private DmTipoFallecimiento As String
    Private DmCausaMuerte As String
Private DmGenero As String
Private DmSqlConec As SqlConnection
#End Region

#Region "Propertys"

    Public Property Genero As String
    Get
        Return DmGenero
    End Get
    Set(ByVal value As String)
        DmGenero = value
    End Set
End Property

    Public Property CausaMuerte As String
    Get
        Return DmCausaMuerte
    End Get
    Set(ByVal value As String)
        DmCausaMuerte = value
    End Set
End Property

    Public Property TipoFallecimiento As String
    Get
        Return DmTipoFallecimiento
    End Get
    Set(ByVal value As String)
        DmTipoFallecimiento = value
    End Set
End Property

    Public Property NumeroPartidaDefuncion() As String
    Get
        Return DmNumeroPartidaDefuncion
    End Get
    Set(ByVal Value As String)
        DmNumeroPartidaDefuncion = Value
    End Set
End Property

    Public Property CodIsam() As Integer
    Get
        Return DmCodIsam
    End Get
    Set(ByVal Value As Integer)
        DmCodIsam = Value
    End Set
End Property

```

```

        End Set
    End Property

    Public Property NombreSolicitante() As String
        Get
            Return DmNombreSolicitante
        End Get
        Set(ByVal Value As String)
            DmNombreSolicitante = Value
        End Set
    End Property

    Public Property NombreFallecido() As String
        Get
            Return DmNombreFallecido
        End Get
        Set(ByVal Value As String)
            DmNombreFallecido = Value
        End Set
    End Property

    Public Property Edad() As Integer
        Get
            Return DmEdad
        End Get
        Set(ByVal Value As Integer)
            DmEdad = Value
        End Set
    End Property

    Public Property EstadoCivil() As String
        Get
            Return DmEstadoCivil
        End Get
        Set(ByVal Value As String)
            DmEstadoCivil = Value
        End Set
    End Property

    Public Property OrigenFallecido() As String
        Get
            Return DmOrigenFallecido
        End Get
        Set(ByVal Value As String)
            DmOrigenFallecido = Value
        End Set
    End Property

    Public Property DireccionFallecido() As String
        Get
            Return DmDireccionFallecido
        End Get
        Set(ByVal Value As String)
            DmDireccionFallecido = Value
        End Set
    End Property

```



```

Public Property OficioProfesionFallecidos() As String
    Get
        Return DmOficioProfesionFallecidos
    End Get
    Set(ByVal Value As String)
        DmOficioProfesionFallecidos = Value
    End Set
End Property

Public Property NombrePadre() As String
    Get
        Return DmNombrePadre
    End Get
    Set(ByVal Value As String)
        DmNombrePadre = Value
    End Set
End Property

Public Property NombreMadre() As String
    Get
        Return DmNombreMadre
    End Get
    Set(ByVal Value As String)
        DmNombreMadre = Value
    End Set
End Property

Public Property MedicoAvalaDefuncion() As String
    Get
        Return DmMedicoAvalaDefuncion
    End Get
    Set(ByVal Value As String)
        DmMedicoAvalaDefuncion = Value
    End Set
End Property

Public Property HoraFechaFallecimiento() As DateTime
    Get
        Return DmHoraFechaFallecimiento
    End Get
    Set(ByVal Value As DateTime)
        DmHoraFechaFallecimiento = Value
    End Set
End Property

Public Property CodEmpleado() As Integer
    Get
        Return DmCodEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodEmpleado = Value
    End Set
End Property

Public Property GetConexion As SqlConnection
    Get
        Return DmSqlConec
    End Get

```

```

        End Get
        Set(ByVal value As SqlConnection)
            DmSqlConec = value
        End Set
    End Property
#End Region

#Region "Funciones"

    'Devuelve el detalle de un isam para los reportes
    Public Function DevolverDetalleIsam(ByVal NumeroIsam As Integer, ByVal CodIsam
As Integer, ByVal Isdem As Integer, ByVal SerieFactura As String) As DataSet
        Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARDETALLEISAM", DmSqlConec)
        Dim ds As New DataSet

        adp.SelectCommand.Parameters.Add("@CodIsam", SqlDbType.Int).Value = CodIsam
'este se refiere al autonumerico de control y no al numero de la isam
        adp.SelectCommand.Parameters.Add("@NumeroIsam", SqlDbType.Int).Value = Isdem
        adp.SelectCommand.Parameters.Add("@Isdem", SqlDbType.Int).Value = Isdem
        adp.SelectCommand.Parameters.Add("@Serie", SqlDbType.VarChar, 2).Value =
SerieFactura
        Try
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlConec.Close()
        End Try
        Return ds
    End Function

    Public Function VerificarPartida() As Boolean 'Verifica la existencia de una
partida de defuncion

        Dim cmd As New SqlCommand("MODRIN.SP_VERIFICARPATIDAEFUNCION", DmSqlConec)
        Dim data As SqlDataReader

        Try
            cmd.CommandType = CommandType.StoredProcedure
            cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
            cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar,
6).Value = DmNumeroPartidaDefuncion
            data = cmd.ExecuteReader()

            If data.Read Then
                Return True
            Else
                Return False
            End If
        Catch
            Throw
        Finally
            cmd.Dispose()

```

```

        DmSqlConec.Close()
    End Try
End Function

Public Function ListarDefunciones() As DataSet ' Muestra todas las partidas de
defuncion
    Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARDEFUNCIONES", DmSqlConec)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
    End Try
    Return ds
End Function

Public Sub BuscarPartidaDefuncion()

    Dim cmd As New SqlCommand("MODRIN.SP_BUSCARDEFUNCIONCODIGO", DmSqlConec)
    Dim data As SqlDataReader

    Try
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()

        If data.Read Then
            If Not IsDBNull(data.Item("NumeroPartidaDefuncion")) Then
DmNumeroPartidaDefuncion = data.Item("NumeroPartidaDefuncion")
                If Not IsDBNull(data.Item("CodIsam")) Then DmCodIsam =
data.Item("CodIsam")
                    If Not IsDBNull(data.Item("NombreSolicitante")) Then
DmNombreSolicitante = data.Item("NombreSolicitante")
                        If Not IsDBNull(data.Item("NombreFallecido")) Then DmNombreFallecido
= data.Item("NombreFallecido")
                            If Not IsDBNull(data.Item("Edad")) Then DmEdad = data.Item("Edad")
                                If Not IsDBNull(data.Item("EstadoCivil")) Then DmEstadoCivil =
data.Item("EstadoCivil")
                                    If Not IsDBNull(data.Item("OrigenFallecido")) Then DmOrigenFallecido
= data.Item("OrigenFallecido")
                                        If Not IsDBNull(data.Item("DireccionFallecido")) Then
DmDireccionFallecido = data.Item("DireccionFallecido")
                                            If Not IsDBNull(data.Item("OficioProfesionFallecidos")) Then
DmOficioProfesionFallecidos = data.Item("OficioProfesionFallecidos")
                                                If Not IsDBNull(data.Item("NombrePadre")) Then DmNombrePadre =
data.Item("NombrePadre")
                                                    If Not IsDBNull(data.Item("NombreMadre")) Then DmNombreMadre =
data.Item("NombreMadre")
                                                        If Not IsDBNull(data.Item("MedicoAvalaDefuncion")) Then
DmMedicoAvalaDefuncion = data.Item("MedicoAvalaDefuncion")
                                                            If Not IsDBNull(data.Item("HoraFechaFallecimiento")) Then
DmHoraFechaFallecimiento = data.Item("HoraFechaFallecimiento")
                                                                If Not IsDBNull(data.Item("CodEmpleado")) Then DmCodEmpleado =
data.Item("CodEmpleado")

```

```

                If Not IsDBNull(data.Item("CausaMuerte")) Then DmCausaMuerte =
data.Item("CausaMuerte")
                If Not IsDBNull(data.Item("Genero")) Then DmGenero =
data.Item("Genero")
            End If

        Catch
            Throw
        Finally
            cmd.Dispose()
        End Try

    End Sub

    Public Function GuardarPartidaDefuncion() As Boolean

        Dim cmd As New SqlCommand("MODRIN.SP_GUARDARPARTIDADEFUNCION", DmSqlConec)

        cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
        cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
        cmd.Parameters.Add("@NombreSolicitante", SqlDbType.VarChar, 250).Value =
DmNombreSolicitante
        cmd.Parameters.Add("@NombreFallecido", SqlDbType.VarChar, 250).Value =
DmNombreFallecido
        cmd.Parameters.Add("@Edad", SqlDbType.Int).Value = DmEdad
        cmd.Parameters.Add("@EstadoCivil", SqlDbType.VarChar, 1).Value =
DmEstadoCivil
        cmd.Parameters.Add("@OrigenFallecido", SqlDbType.VarChar, 50).Value =
DmOrigenFallecido
        cmd.Parameters.Add("@DireccionFallecido", SqlDbType.VarChar, 350).Value =
DmDireccionFallecido
        cmd.Parameters.Add("@OficioProfesionFallecidos", SqlDbType.VarChar,
250).Value = DmOficioProfesionFallecidos
        cmd.Parameters.Add("@NombrePadre", SqlDbType.VarChar, 250).Value =
DmNombrePadre
        cmd.Parameters.Add("@NombreMadre", SqlDbType.VarChar, 250).Value =
DmNombreMadre
        cmd.Parameters.Add("@MedicoAvalaDefuncion", SqlDbType.VarChar, 250).Value =
DmMedicoAvalaDefuncion
        cmd.Parameters.Add("@HoraFechaFallecimiento", SqlDbType.DateTime).Value =
DmHoraFechaFallecimiento
        cmd.Parameters.Add("@TipoFallecimiento", SqlDbType.VarChar, 50).Value =
DmTipoFallecimiento
        cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
        cmd.Parameters.Add("@CausaMuerte", SqlDbType.VarChar, 150).Value =
DmCausaMuerte
        cmd.Parameters.Add("@Genero", SqlDbType.VarChar, 12).Value = DmGenero
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarPartidaDefuncion = cmd.ExecuteNonQuery
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

```

```

End Function

Public Function ModificarPartidaDefuncion() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_MODIFICARPARTIDADEFUNCION", DmSqlConec)

    cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
    cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
    cmd.Parameters.Add("@NombreSolicitante", SqlDbType.VarChar, 250).Value =
DmNombreSolicitante
    cmd.Parameters.Add("@NombreFallecido", SqlDbType.VarChar, 250).Value =
DmNombreFallecido
    cmd.Parameters.Add("@Edad", SqlDbType.Int).Value = DmEdad
    cmd.Parameters.Add("@EstadoCivil", SqlDbType.VarChar, 1).Value =
DmEstadoCivil
    cmd.Parameters.Add("@OrigenFallecido", SqlDbType.VarChar, 50).Value =
DmOrigenFallecido
    cmd.Parameters.Add("@DireccionFallecido", SqlDbType.VarChar, 350).Value =
DmDireccionFallecido
    cmd.Parameters.Add("@OficioProfesionFallecidos", SqlDbType.VarChar,
250).Value = DmOficioProfesionFallecidos
    cmd.Parameters.Add("@NombrePadre", SqlDbType.VarChar, 250).Value =
DmNombrePadre
    cmd.Parameters.Add("@NombreMadre", SqlDbType.VarChar, 250).Value =
DmNombreMadre
    cmd.Parameters.Add("@MedicoAvalaDefuncion", SqlDbType.VarChar, 250).Value =
DmMedicoAvalaDefuncion
    cmd.Parameters.Add("@HoraFechaFallecimiento", SqlDbType.DateTime).Value =
DmHoraFechaFallecimiento
    cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
    cmd.Parameters.Add("@CausaMuerte", SqlDbType.VarChar, 150).Value =
DmCausaMuerte
    cmd.Parameters.Add("@Genero", SqlDbType.VarChar, 12).Value = DmGenero
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarPartidaDefuncion = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Function
#End Region
End Class

```

CIPropiedad: Clase encargada de gestionar todas las transacciones realizadas con las propiedades que se venden a los clientes del Cementerio Santa Isabel.

```

Imports System.Data.SqlClient

Public Class CIPropiedad
#Region "Variables"
    Private DmCodPropiead As Integer

```

```

Private DmTipoPropiedad As String
Private DmCodIsam As Integer
Private DmCodTitulo As Integer
Private DmCodCliente As Integer
Private DmCodEmpleado As Integer
Private DmNombresComprador As String
Private DmApellidosComprador As String
Private DmOficioProfesionComprador As String
Private DmTelefonoComprador As String
Private DmDireccionComprador As String
Private DmNumeroDuiComprador As String
Private DmLugarExtesnsionDuiComprador As String
Private DmFechaExtensionDuiComprador As DateTime
Private DmZona As String
Private DmCategoria As Integer
Private DmCuadro As String
Private DmCalle As String
Private DmPuestoSepultura As String
Private DmFila As String
Private DmAnchoPropiead As String
Private DmLargoPropiead As String
Private DmNumeroPuestoNorte As Integer
Private DmNumeroCalleNorte As Integer
Private DmNumeroPuestoSur As Integer
Private DmNumeroCalleSur As Integer
Private DmNumeroPuestoOriente As Integer
Private DmNumeroCalleOriente As Integer
Private DmNumeroPuestoPoniente As Integer
Private DmNumeroCallePoniente As Integer
Private DmNumeroNichosConstruir As Integer
Private DmFechaCompra As DateTime
Private DmNula As Boolean
Private DmFechaNulidad As DateTime
Private DmUsuarioAnulador As Integer
Private DmUsuarioAutoriza As Integer
Private DmJustificacionNulidad As String
Private DmNoActa As Integer
Private DmDptoExtDui As String
Private DmFirmas As Integer
Private DmNoCalle As String
Private DmLetraCalle As String
Private DmSqlCon As SqlConnection

```

#End Region

#Region "Propertys"

```

Public Property NoCalle As String
    Get
        Return DmNoCalle
    End Get
    Set(ByVal value As String)
        DmNoCalle = value
    End Set
End Property

Public Property LetraCalle As String

```

```

    Get
        Return DmLetraCalle
    End Get
    Set(ByVal value As String)
        DmLetraCalle = value
    End Set
End Property

Public Property Firmas As Integer
    Get
        Return DmFirmas
    End Get
    Set(ByVal value As Integer)
        DmFirmas = value
    End Set
End Property

Public Property DptoExtDui As String
    Get
        Return DmDptoExtDui
    End Get
    Set(ByVal value As String)
        DmDptoExtDui = value
    End Set
End Property

Public Property NoActa As Integer
    Get
        Return DmNoActa
    End Get
    Set(ByVal value As Integer)
        DmNoActa = value
    End Set
End Property

Public Property Nula As Boolean
    Get
        Return DmNula
    End Get
    Set(ByVal value As Boolean)
        DmNula = value
    End Set
End Property

Public Property FechaNulidad As DateTime
    Get
        Return DmFechaNulidad
    End Get
    Set(ByVal value As DateTime)
        DmFechaNulidad = value
    End Set
End Property

Public Property UsuarioAnulador As Integer
    Get
        Return DmUsuarioAnulador
    End Get

```

```

        Set(ByVal value As Integer)
            DmUsuarioAnulador = value
        End Set
    End Property

    Public Property UsuarioAutorizador As Integer
        Get
            Return DmUsuarioAutoriza
        End Get
        Set(ByVal value As Integer)
            DmUsuarioAutoriza = value
        End Set
    End Property

    Public Property JustificacionNulidad As String
        Get
            Return DmJustificacionNulidad
        End Get
        Set(ByVal value As String)
            DmJustificacionNulidad = value
        End Set
    End Property

    Public Property CodPropiead() As Integer
        Get
            Return DmCodPropiead
        End Get
        Set(ByVal Value As Integer)
            DmCodPropiead = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)
            DmSqlCon = Value
        End Set
    End Property

    Public Property TipoPropiedad() As String
        Get
            Return DmTipoPropiedad
        End Get
        Set(ByVal Value As String)
            DmTipoPropiedad = Value
        End Set
    End Property

    Public Property CodIsam() As Integer
        Get
            Return DmCodIsam
        End Get
        Set(ByVal Value As Integer)
            DmCodIsam = Value
        End Set
    End Property

```



```

End Property

Public Property CodTitulo() As Integer
    Get
        Return DmCodTitulo
    End Get
    Set(ByVal Value As Integer)
        DmCodTitulo = Value
    End Set
End Property

Public Property CodCliente() As Integer
    Get
        Return DmCodCliente
    End Get
    Set(ByVal Value As Integer)
        DmCodCliente = Value
    End Set
End Property

Public Property CodEmpleado() As Integer
    Get
        Return DmCodEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodEmpleado = Value
    End Set
End Property

Public Property NombresComprador() As String
    Get
        Return DmNombresComprador
    End Get
    Set(ByVal Value As String)
        DmNombresComprador = Value
    End Set
End Property

Public Property ApellidosComprador() As String
    Get
        Return DmApellidosComprador
    End Get
    Set(ByVal Value As String)
        DmApellidosComprador = Value
    End Set
End Property

Public Property OficioProfesionComprador() As String
    Get
        Return DmOficioProfesionComprador
    End Get
    Set(ByVal Value As String)
        DmOficioProfesionComprador = Value
    End Set
End Property

Public Property TelefonoComprador() As String

```

```

    Get
        Return DmTelefonoComprador
    End Get
    Set(ByVal Value As String)
        DmTelefonoComprador = Value
    End Set
End Property

Public Property DireccionComprador() As String
    Get
        Return DmDireccionComprador
    End Get
    Set(ByVal Value As String)
        DmDireccionComprador = Value
    End Set
End Property

Public Property NumeroDuiComprador() As String
    Get
        Return DmNumeroDuiComprador
    End Get
    Set(ByVal Value As String)
        DmNumeroDuiComprador = Value
    End Set
End Property

Public Property LugarExtesnsionDuiComprador() As String
    Get
        Return DmLugarExtesnsionDuiComprador
    End Get
    Set(ByVal Value As String)
        DmLugarExtesnsionDuiComprador = Value
    End Set
End Property

Public Property FechaExtensionDuiComprador() As DateTime
    Get
        Return DmFechaExtensionDuiComprador
    End Get
    Set(ByVal Value As DateTime)
        DmFechaExtensionDuiComprador = Value
    End Set
End Property

Public Property Zona() As String
    Get
        Return DmZona
    End Get
    Set(ByVal Value As String)
        DmZona = Value
    End Set
End Property

Public Property Categoria() As Integer
    Get
        Return DmCategoria
    End Get

```

```

        Set(ByVal Value As Integer)
            DmCategoria = Value
        End Set
    End Property

    Public Property Cuadro() As String
        Get
            Return DmCuadro
        End Get
        Set(ByVal Value As String)
            DmCuadro = Value
        End Set
    End Property

    Public Property Calle() As String
        Get
            Return DmCalle
        End Get
        Set(ByVal Value As String)
            DmCalle = Value
        End Set
    End Property

    Public Property PuestoSepultura() As String
        Get
            Return DmPuestoSepultura
        End Get
        Set(ByVal Value As String)
            DmPuestoSepultura = Value
        End Set
    End Property

    Public Property Fila() As String
        Get
            Return DmFila
        End Get
        Set(ByVal Value As String)
            DmFila = Value
        End Set
    End Property

    Public Property AnchoPropiedad() As String
        Get
            Return DmAnchoPropiead
        End Get
        Set(ByVal Value As String)
            DmAnchoPropiead = Value
        End Set
    End Property

    Public Property LargoPropiedad() As String
        Get
            Return DmLargoPropiedad
        End Get
        Set(ByVal Value As String)
            DmLargoPropiedad = Value
        End Set
    End Property

```

```

End Property

Public Property NumeroPuestoNorte() As Integer
    Get
        Return DmNumeroPuestoNorte
    End Get
    Set(ByVal Value As Integer)
        DmNumeroPuestoNorte = Value
    End Set
End Property

Public Property NumeroCalleNorte() As Integer
    Get
        Return DmNumeroCalleNorte
    End Get
    Set(ByVal Value As Integer)
        DmNumeroCalleNorte = Value
    End Set
End Property

Public Property NumeroPuestoSur() As Integer
    Get
        Return DmNumeroPuestoSur
    End Get
    Set(ByVal Value As Integer)
        DmNumeroPuestoSur = Value
    End Set
End Property

Public Property NumeroCalleSur() As Integer
    Get
        Return DmNumeroCalleSur
    End Get
    Set(ByVal Value As Integer)
        DmNumeroCalleSur = Value
    End Set
End Property

Public Property NumeroPuestoOriente() As Integer
    Get
        Return DmNumeroPuestoOriente
    End Get
    Set(ByVal Value As Integer)
        DmNumeroPuestoOriente = Value
    End Set
End Property

Public Property NumeroCalleOriente() As Integer
    Get
        Return DmNumeroCalleOriente
    End Get
    Set(ByVal Value As Integer)
        DmNumeroCalleOriente = Value
    End Set
End Property

Public Property NumeroPuestoPoniente() As Integer

```

```

        Get
            Return DmNumeroPuestoPoniente
        End Get
        Set(ByVal Value As Integer)
            DmNumeroPuestoPoniente = Value
        End Set
    End Property

    Public Property NumeroCallePoniente() As Integer
        Get
            Return DmNumeroCallePoniente
        End Get
        Set(ByVal Value As Integer)
            DmNumeroCallePoniente = Value
        End Set
    End Property

    Public Property NumeroNichosConstruir() As Integer
        Get
            Return DmNumeroNichosConstruir
        End Get
        Set(ByVal Value As Integer)
            DmNumeroNichosConstruir = Value
        End Set
    End Property

    Public Property FechaCompra() As DateTime
        Get
            Return DmFechaCompra
        End Get
        Set(ByVal Value As DateTime)
            DmFechaCompra = Value
        End Set
    End Property
#End Region

#Region "Funciones"

    Public Function ValorTipoNicho(ByVal TipoNicho As String) As Double
        Dim cmd As New SqlCommand("MODRIN.SP_VALORTIPONICHO", DmSqlCon)
        Dim data As SqlDataReader
        Dim CostoNichoFrm As Double = 0.0
        Try
            cmd.CommandType = CommandType.StoredProcedure
            cmd.Parameters.Add("@TipoNicho", SqlDbType.VarChar, 20).Value =
TipoNicho

            data = cmd.ExecuteReader()
            If data.Read Then
                If Not IsDBNull(data.Item("TotalServicio")) Then CostoNichoFrm =
data.Item("TotalServicio")
                Return CostoNichoFrm
            Else
                Return 0
            End If
        Catch

```

```

        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ValorFormulario() As Double
    Dim cmd As New SqlCommand("MODRIN.SP_VALORFRMNICHO", DmSqlCon)
    Dim data As SqlDataReader
    Dim CostoFrm As Double = 0.0
    Try
        cmd.CommandType = CommandType.StoredProcedure

        data = cmd.ExecuteReader()
        If data.Read Then
            If Not IsDBNull(data.Item("ValorFormulario")) Then CostoFrm =
data.Item("ValorFormulario")
                Return CostoFrm
            Else
                Return 0
            End If
        End If

    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ValorFF() As Double
    Dim cmd As New SqlCommand("MODRIN.SP_VALORFF", DmSqlCon)
    Dim data As SqlDataReader
    Dim CostoFF As Double = 0.0
    Try
        cmd.CommandType = CommandType.StoredProcedure

        data = cmd.ExecuteReader()
        If data.Read Then
            If Not IsDBNull(data.Item("FF")) Then CostoFF = data.Item("FF")
                Return CostoFF
            Else
                Return 0
            End If
        End If

    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function AnularTitulo() As Boolean

```

```

Dim cmd As New SqlCommand("MODRIN.SP_ANULARTITULOPROPIEDAD", DmSqlCon)

cmd.Parameters.Add("@Nula", SqlDbType.Bit).Value = DmNula
cmd.Parameters.Add("@FechaNulidad", SqlDbType.DateTime).Value =
DmFechaNulidad
cmd.Parameters.Add("@UsuarioAnulador", SqlDbType.Int).Value =
DmUsuarioAnulador
cmd.Parameters.Add("@UsuarioAutoriza", SqlDbType.Int).Value =
DmUsuarioAutoriza
cmd.Parameters.Add("@JustificacionNulidad", SqlDbType.VarChar, 250).Value =
DmJustificacionNulidad
cmd.Parameters.Add("@CodPropiedad", SqlDbType.Int).Value = DmCodPropiead
cmd.Parameters.Add("@TipoPropiedad", SqlDbType.VarChar, 50).Value =
DmTipoPropiedad

Try
    cmd.CommandType = CommandType.StoredProcedure
    AnularTitulo = cmd.ExecuteNonQuery()
Catch
    Throw
Finally
    cmd.Dispose()
End Try
End Function

Public Function ListarPropiead() As DataSet ' Muestra todas las propiedades con
y sin fallecidos
Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARPROPIEDAD", DmSqlCon)
Dim ds As New DataSet
Try
    adp.SelectCommand.CommandType = CommandType.StoredProcedure
    adp.Fill(ds)
Catch
    Throw
Finally
    adp.Dispose()
End Try
Return ds
End Function

Public Function ListarPropieadFallecido() As DataSet ' Muestra todas las
propiedades con fallecidos
Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARPROPIEDADFALLECIDOS",
DmSqlCon)
Dim ds As New DataSet
Try
    adp.SelectCommand.CommandType = CommandType.StoredProcedure
    adp.Fill(ds)
Catch
    Throw
Finally
    adp.Dispose()
End Try
Return ds
End Function

Public Function VerificarPropiedad() As Boolean

```

```

Dim cmd As New SqlCommand("MODRIN.SP_VERIFICARPROPIEDAD", DmSqlCon)
Dim data As SqlDataReader

Try
    cmd.CommandType = CommandType.StoredProcedure
    cmd.Parameters.Add("@TipoPropiedad", SqlDbType.VarChar, 50).Value =
DmTipoPropiedad
    cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
    cmd.Parameters.Add("@CodTitulo", SqlDbType.Int).Value = DmCodTitulo
    data = cmd.ExecuteReader()

    If data.Read Then
        Return True
    Else
        Return False
    End If
Catch
    Throw
Finally
    cmd.Dispose()
    DmSqlCon.Close()
End Try
End Function

Public Sub BuscarCodigo() ' Obtiene el codigo de la propiedad en base a
ciertos parametros
Dim cmd As New SqlCommand("MODRIN.SP_OBTENERCODIGOPROPIEDAD", DmSqlCon)
Dim data As SqlDataReader
Try
    cmd.CommandType = CommandType.StoredProcedure
    cmd.Parameters.Add("@Cliente", SqlDbType.Int).Value = DmCodCliente
    cmd.Parameters.Add("@ISAM", SqlDbType.Int).Value = DmCodIsam
    cmd.Parameters.Add("@Titulo", SqlDbType.Int).Value = DmCodTitulo
    data = cmd.ExecuteReader()
    If data.Read Then
        If Not IsDBNull(data.Item("CodPropiead")) Then DmCodPropiead =
data.Item("CodPropiead")
    End If
Catch
    Throw
Finally
    cmd.Dispose()
    DmSqlCon.Dispose()
End Try
End Sub

Public Function ListarPropieadBeneficiario() As DataSet ' Muestra todas las
propiedades con beneficiarios
Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARPROPIEDADBENEFICIARIOS",
DmSqlCon)
Dim ds As New DataSet
Try
    adp.SelectCommand.CommandType = CommandType.StoredProcedure
    adp.Fill(ds)
Catch

```



```

        Throw
    Finally
        adp.Dispose()
    End Try
    Return ds
End Function

Public Function BuscarPropiedadCodigo(ByVal _CodigoTitulo As Integer, ByVal
_TipoPropiedad As String) As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_BUSCARPROPIEDADCODIGO", DmSqlCon)
    Dim data As SqlDataReader

    Try
        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@CodTitulo", SqlDbType.Int).Value = _CodigoTitulo
        cmd.Parameters.Add("@TipoPropiedad", SqlDbType.VarChar, 50).Value =
_TipoPropiedad
        data = cmd.ExecuteReader()

        If data.Read Then
            If Not IsDBNull(data.Item("CodPropiead")) Then DmCodPropiead =
data.Item("CodPropiead")
            If Not IsDBNull(data.Item("TipoPropiedad")) Then DmTipoPropiedad =
data.Item("TipoPropiedad")
            If Not IsDBNull(data.Item("CodIsam")) Then DmCodIsam =
data.Item("CodIsam")
            If Not IsDBNull(data.Item("CodTitulo")) Then DmCodTitulo =
data.Item("CodTitulo")
            If Not IsDBNull(data.Item("CodActa")) Then DmNoActa =
data.Item("CodActa")
            If Not IsDBNull(data.Item("CodCliente")) Then DmCodCliente =
data.Item("CodCliente")
            If Not IsDBNull(data.Item("CodEmpleado")) Then DmCodEmpleado =
data.Item("CodEmpleado")
            If Not IsDBNull(data.Item("NombresComprador")) Then
DmNombresComprador = data.Item("NombresComprador")
            If Not IsDBNull(data.Item("ApellidosComprador")) Then
DmApellidosComprador = data.Item("ApellidosComprador")
            If Not IsDBNull(data.Item("OficioProfesionComprador")) Then
DmOficioProfesionComprador = data.Item("OficioProfesionComprador")
            If Not IsDBNull(data.Item("TelefonoComprador")) Then
DmTelefonoComprador = data.Item("TelefonoComprador")
            If Not IsDBNull(data.Item("DireccionComprador")) Then
DmDireccionComprador = data.Item("DireccionComprador")
            If Not IsDBNull(data.Item("NumeroDuiComprador")) Then
DmNumeroDuiComprador = data.Item("NumeroDuiComprador")
            If Not IsDBNull(data.Item("LugarExtesnsionDuiComprador")) Then
DmLugarExtesnsionDuiComprador = data.Item("LugarExtesnsionDuiComprador")
            If Not IsDBNull(data.Item("FechaExtensionDuiComprador")) Then
DmFechaExtensionDuiComprador = data.Item("FechaExtensionDuiComprador")
            If Not IsDBNull(data.Item("Zona")) Then DmZona = data.Item("Zona")
            If Not IsDBNull(data.Item("Categoria")) Then DmCategoria =
data.Item("Categoria")
            If Not IsDBNull(data.Item("Cuadro")) Then DmCuadro =
data.Item("Cuadro")

```

```

        If Not IsDBNull(data.Item("Calle")) Then DmCalle =
data.Item("Calle")
        If Not IsDBNull(data.Item("NoCalle")) Then DmNoCalle =
data.Item("NoCalle")
        If Not IsDBNull(data.Item("LetraCalle")) Then DmLetraCalle =
data.Item("LetraCalle")
        If Not IsDBNull(data.Item("PuestoSepultura")) Then DmPuestoSepultura
= data.Item("PuestoSepultura")
        If Not IsDBNull(data.Item("Fila")) Then DmFila = data.Item("Fila")
        If Not IsDBNull(data.Item("AnchoPropiead")) Then DmAnchoPropiead =
data.Item("AnchoPropiead")
        If Not IsDBNull(data.Item("LargoPropiedad")) Then DmLargoPropiedad
= data.Item("LargoPropiedad")
        If Not IsDBNull(data.Item("NumeroPuestoNorte")) Then
DmNumeroPuestoNorte = data.Item("NumeroPuestoNorte")
        If Not IsDBNull(data.Item("NumeroCalleNorte")) Then
DmNumeroCalleNorte = data.Item("NumeroCalleNorte")
        If Not IsDBNull(data.Item("NumeroPuestoSur")) Then DmNumeroPuestoSur
= data.Item("NumeroPuestoSur")
        If Not IsDBNull(data.Item("NumeroCalleSur")) Then DmNumeroCalleSur =
data.Item("NumeroCalleSur")
        If Not IsDBNull(data.Item("NumeroPuestoOriente")) Then
DmNumeroPuestoOriente = data.Item("NumeroPuestoOriente")
        If Not IsDBNull(data.Item("NumeroCalleOriente")) Then
DmNumeroCalleOriente = data.Item("NumeroCalleOriente")
        If Not IsDBNull(data.Item("NumeroPuestoPoniente")) Then
DmNumeroPuestoPoniente = data.Item("NumeroPuestoPoniente")
        If Not IsDBNull(data.Item("NumeroCallePoniente")) Then
DmNumeroCallePoniente = data.Item("NumeroCallePoniente")
        If Not IsDBNull(data.Item("NumeroNichosConstruir")) Then
DmNumeroNichosConstruir = data.Item("NumeroNichosConstruir")
        If Not IsDBNull(data.Item("FechaCompra")) Then DmFechaCompra =
data.Item("FechaCompra")
        If Not IsDBNull(data.Item("DptoExtDui")) Then DmDptoExtDui =
data.Item("DptoExtDui")
        If Not IsDBNull(data.Item("CodFirmas")) Then DmFirmas =
data.Item("CodFirmas")
        BuscarPropiedadCodigo = True
    Else
        BuscarPropiedadCodigo = False
    End If
Catch
    Throw
Finally
    cmd.Dispose()
End Try

End Function

Public Function GuardarPropiedad() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_GUARDARPROPIEDAD", DmSqlCon)

    cmd.Parameters.Add("@TipoPropiedad", SqlDbType.VarChar, 50).Value =
DmTipoPropiedad
    cmd.Parameters.Add("@CodIsam", SqlDbType.Int).Value = DmCodIsam
    cmd.Parameters.Add("@CodTitulo", SqlDbType.Int).Value = DmCodTitulo

```

```

cmd.Parameters.Add("@CodCliente", SqlDbType.Int).Value = DmCodCliente
cmd.Parameters.Add("@CodActa", SqlDbType.Int).Value = DmNoActa
cmd.Parameters.Add("@CodEmpleado", SqlDbType.Int).Value = DmCodEmpleado
cmd.Parameters.Add("@NombresComprador", SqlDbType.VarChar, 250).Value =
DmNombresComprador
cmd.Parameters.Add("@ApellidosComprador", SqlDbType.VarChar, 250).Value =
DmApellidosComprador
cmd.Parameters.Add("@OficioProfesionComprador", SqlDbType.VarChar,
250).Value = DmOficioProfesionComprador
cmd.Parameters.Add("@TelefonoComprador", SqlDbType.VarChar, 9).Value =
DmTelefonoComprador
cmd.Parameters.Add("@DireccionComprador", SqlDbType.VarChar, 350).Value =
DmDireccionComprador
cmd.Parameters.Add("@NumeroDuiComprador", SqlDbType.VarChar, 10).Value =
DmNumeroDuiComprador
cmd.Parameters.Add("@LugarExtesnsionDuiComprador", SqlDbType.VarChar,
50).Value = DmLugarExtesnsionDuiComprador
cmd.Parameters.Add("@FechaExtensionDuiComprador", SqlDbType.DateTime).Value
= DmFechaExtensionDuiComprador
cmd.Parameters.Add("@Zona", SqlDbType.VarChar, 50).Value = DmZona
cmd.Parameters.Add("@Categoria", SqlDbType.Int).Value = DmCategoria
cmd.Parameters.Add("@Cuadro", SqlDbType.VarChar, 50).Value = DmCuadro
cmd.Parameters.Add("@Calle", SqlDbType.VarChar, 50).Value = DmCalle
cmd.Parameters.Add("@NoCalle", SqlDbType.VarChar, 10).Value = DmNoCalle
cmd.Parameters.Add("@LetraCalle", SqlDbType.VarChar, 10).Value =
DmLetraCalle
cmd.Parameters.Add("@PuestoSepultura", SqlDbType.Int).Value =
DmPuestoSepultura
cmd.Parameters.Add("@Fila", SqlDbType.Int).Value = DmFila
cmd.Parameters.Add("@AnchoPropiead", SqlDbType.Float).Value =
DmAnchoPropiead
cmd.Parameters.Add("@LargoPropieadad", SqlDbType.Float).Value =
DmLargoPropieadad
cmd.Parameters.Add("@NumeroPuestoNorte", SqlDbType.Int).Value =
DmNumeroPuestoNorte
cmd.Parameters.Add("@NumeroCalleNorte", SqlDbType.Int).Value =
DmNumeroCalleNorte
cmd.Parameters.Add("@NumeroPuestoSur", SqlDbType.Int).Value =
DmNumeroPuestoSur
cmd.Parameters.Add("@NumeroCalleSur", SqlDbType.Int).Value =
DmNumeroCalleSur
cmd.Parameters.Add("@NumeroPuestoOriente", SqlDbType.Int).Value =
NumeroPuestoOriente
cmd.Parameters.Add("@NumeroCalleOriente", SqlDbType.Int).Value =
DmNumeroCalleOriente
cmd.Parameters.Add("@NumeroPuestoPoniente", SqlDbType.Int).Value =
DmNumeroPuestoPoniente
cmd.Parameters.Add("@NumeroCallePoniente", SqlDbType.Int).Value =
DmNumeroCallePoniente
cmd.Parameters.Add("@NumeroNichosConstruir", SqlDbType.Int).Value =
DmNumeroNichosConstruir
cmd.Parameters.Add("@FechaCompra", SqlDbType.DateTime).Value = DmFechaCompra
cmd.Parameters.Add("@DptoExtDui", SqlDbType.VarChar, 150).Value =
DmDptoExtDui
cmd.Parameters.Add("@CodFirmas", SqlDbType.Int).Value = DmFirmas

```

Try

```

        cmd.CommandType = CommandType.StoredProcedure
        GuardarPropiedad = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Function
#End Region
End Class

```

CITestigos: Clase encargada de gestionar todas las transacciones realizadas con los testigos de las defunciones que ocurren en las casas de habitación de los fallecidos.

```
Imports System.Data.SqlClient
```

```
Public Class CITestigos
```

```

    Private DmCodTestigo As Integer
    Private DmNumeroPartidaDefuncion As String
    Private DmNombres As String
    Private DmApellidos As String
    Private DmNumeroDui As String
    Private DmParentescoFallecido As String
    Private DmSqlCon As SqlConnection

```

```

Public Property CodTestigo() As Integer
    Get
        Return DmCodTestigo
    End Get
    Set(ByVal Value As Integer)
        DmCodTestigo = Value
    End Set
End Property

```

```

Public Property NumeroPartidaDefuncion() As String
    Get
        Return DmNumeroPartidaDefuncion
    End Get
    Set(ByVal Value As String)
        DmNumeroPartidaDefuncion = Value
    End Set
End Property

```

```

Public Property Nombres() As String
    Get
        Return DmNombres
    End Get
    Set(ByVal Value As String)
        DmNombres = Value
    End Set
End Property

```

```

Public Property Apellidos() As String
    Get

```

```

        Return DmApellidos
    End Get
    Set(ByVal Value As String)
        DmApellidos = Value
    End Set
End Property

Public Property NumeroDui() As String
    Get
        Return DmNumeroDui
    End Get
    Set(ByVal Value As String)
        DmNumeroDui = Value
    End Set
End Property

Public Property ParentescoFallecido() As String
    Get
        Return DmParentescoFallecido
    End Get
    Set(ByVal Value As String)
        DmParentescoFallecido = Value
    End Set
End Property

Public Property GetConexion As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal value As SqlConnection)
        DmSqlCon = value
    End Set
End Property

Public Function ListarTestigos() As DataSet

    Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARTESTIGOS", DmSqlCon)
    Dim ds As New DataSet

    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
    End Try

    Return ds

End Function

Public Sub BuscarTestigo()

    Dim cmd As New SqlCommand("MODRIN.SP_BUSCARTESTIGO", DmSqlCon)
    Dim data As SqlDataReader

```

```

Try
    cmd.CommandType = CommandType.StoredProcedure
    data = cmd.ExecuteReader()

    If data.Read Then
        If Not IsDBNull(data.Item("CodTestigo")) Then DmCodTestigo =
data.Item("CodTestigo")
        If Not IsDBNull(data.Item("NumeroPartidaDefuncion")) Then
DmNumeroPartidaDefuncion = data.Item("NumeroPartidaDefuncion")
        If Not IsDBNull(data.Item("Nombres")) Then DmNombres =
data.Item("Nombres")
        If Not IsDBNull(data.Item("Apellidos")) Then DmApellidos =
data.Item("Apellidos")
        If Not IsDBNull(data.Item("NumeroDui")) Then DmNumeroDui =
data.Item("NumeroDui")
        If Not IsDBNull(data.Item("ParentescoFallecido")) Then
DmParentescoFallecido = data.Item("ParentescoFallecido")
        End If

    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Sub

Public Function GuardarTestigo() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_GUARDARTESTIGO", DmSqlCon)

    cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
    cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 10).Value = DmNumeroDui
    cmd.Parameters.Add("@ParentescoFallecido", SqlDbType.VarChar, 50).Value =
DmParentescoFallecido

    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarTestigo = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Function

Public Function ModificarTestigo() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_MODIFICARTESTIGO", DmSqlCon)

    cmd.Parameters.Add("@CodTestigo", SqlDbType.Int).Value = DmCodTestigo

```

```

        cmd.Parameters.Add("@NumeroPartidaDefuncion", SqlDbType.VarChar, 6).Value =
DmNumeroPartidaDefuncion
        cmd.Parameters.Add("@Nombres", SqlDbType.VarChar, 250).Value = DmNombres
        cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
        cmd.Parameters.Add("@NumeroDui", SqlDbType.VarChar, 10).Value = DmNumeroDui
        cmd.Parameters.Add("@ParentescoFallecido", SqlDbType.VarChar, 50).Value =
DmParentescoFallecido

    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarTestigo = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Function

Public Function EliminarTestigo() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_ELIMINARTESTIGO", DmSqlCon)

    Try
        cmd.CommandType = CommandType.StoredProcedure
        EliminarTestigo = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
    End Try

End Function

End Class

```

CIFirmas: Clase encargada de gestionar las firmas utilizadas en los títulos de propiedad y actas de compras de nichos o fosas.

```

Imports System.Data.SqlClient
Public Class CIFirmas

    Private DmCodFirmas As Integer
    Private DmFirmaAlcalde As String
    Private DmFirmaSecretario As String
    Private DmFirmaAdministrador As String
    Private DmFechaIngreso As DateTime
    Private DmCodUsuario As Integer
    Private DmActiva As Boolean
    Private DmSqlCon As SqlConnection

    Public Property GetConexion As SqlConnection
    Get
        Return DmSqlCon
    End Get

```

```

        Set(ByVal value As SqlConnection)
            DmSqlCon = value
        End Set
    End Property

    Public Property CodFirmas As Integer
        Get
            Return DmCodFirmas
        End Get
        Set(ByVal value As Integer)
            DmCodFirmas = value
        End Set
    End Property

    Public Property FirmaAlcalde As String
        Get
            Return DmFirmaAlcalde
        End Get
        Set(ByVal value As String)
            DmFirmaAlcalde = value
        End Set
    End Property

    Public Property FirmaSecretario As String
        Get
            Return DmFirmaSecretario
        End Get
        Set(ByVal value As String)
            DmFirmaSecretario = value
        End Set
    End Property

    Public Property FirmaAdministrador As String
        Get
            Return DmFirmaAdministrador
        End Get
        Set(ByVal value As String)
            DmFirmaAdministrador = value
        End Set
    End Property

    Public Property FechaIngreso As DateTime
        Get
            Return DmFechaIngreso
        End Get
        Set(ByVal value As DateTime)
            DmFechaIngreso = value
        End Set
    End Property

    Public Property CodUsuario As Integer
        Get
            Return DmCodUsuario
        End Get
        Set(ByVal value As Integer)
            DmCodUsuario = value
        End Set
    End Property

```



```

        End Set
    End Property

    Public Property Activa As Boolean
        Get
            Return DmActiva
        End Get
        Set(ByVal value As Boolean)
            DmActiva = value
        End Set
    End Property

    Public Function CodigoFirmasActivas() As Integer
        Dim cmd As New SqlCommand("MODRIN.SP_CODFIRMASACTIVAS", DmSqlCon)
        Dim data As SqlDataReader
        Dim codigo As Integer = 0
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()

        If data.Read Then
            If Not IsDBNull(data.Item("CodFirmas")) Then codigo =
data.Item("CodFirmas")
            End If
            Return codigo
        End Function

    Public Function FirmasActivas() As Boolean
        Dim cmd As New SqlCommand("MODRIN.SP_FIRMASACTIVAS", DmSqlCon)
        Dim data As SqlDataReader

        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()

        If data.Read Then
            If Not IsDBNull(data.Item("CodFirmas")) Then DmCodFirmas =
data.Item("CodFirmas")
                If Not IsDBNull(data.Item("FirmaAdministrador")) Then
DmFirmaAdministrador = data.Item("FirmaAdministrador")
                    If Not IsDBNull(data.Item("FirmaAlcalde")) Then DmFirmaAlcalde =
data.Item("FirmaAlcalde")
                        If Not IsDBNull(data.Item("FirmaSecretario")) Then DmFirmaSecretario =
data.Item("FirmaSecretario")
                            If Not IsDBNull(data.Item("FechaIngreso")) Then DmFechaIngreso =
data.Item("FechaIngreso")
                                If Not IsDBNull(data.Item("CodUsuario")) Then DmCodUsuario =
data.Item("CodUsuario")
                                    If Not IsDBNull(data.Item("Activa")) Then DmActiva = data.Item("Activa")
                                        Return True
                                    Else
                                        Return False
                                    End If
                                End If
                            End Function

        Public Function GuardarFirmas() As Boolean

```

```

    Dim cmd As New SqlCommand("MODRIN.SP_GUARDARFIRMAS", DmSqlCon)

    cmd.Parameters.Add("@FirmaAdministrador", SqlDbType.VarChar, 250).Value =
DmFirmaAdministrador
    cmd.Parameters.Add("@FirmaAlcalde", SqlDbType.VarChar, 250).Value =
DmFirmaAlcalde
    cmd.Parameters.Add("@FirmaSecretario", SqlDbType.VarChar, 250).Value =
DmFirmaSecretario
    cmd.Parameters.Add("@FechaIngreso", SqlDbType.DateTime).Value =
DmFechaIngreso
    cmd.Parameters.Add("@CodUsuario", SqlDbType.Int).Value = DmCodUsuario
    cmd.Parameters.Add("@Activa", SqlDbType.Bit).Value = DmActiva

    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarFirmas = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try

End Function

Public Function InhabilitarFirmas() As Boolean

    Dim cmd As New SqlCommand("MODRIN.SP_ANULARFIRMAS", DmSqlCon)

    cmd.Parameters.Add("@CodFirmas", SqlDbType.Int).Value = DmCodFirmas
'cmd.Parameters.Add("@Activa", SqlDbType.Bit).Value = DmActiva

    Try
        cmd.CommandType = CommandType.StoredProcedure
        InhabilitarFirmas = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try

End Function
End Class

```

CLASES DEL MODULO DE PERSONAL (MODPER)

CIDepartamento: encarga de comunicarse con la tabla de departamentos dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient

Public Class CIDepartamento

```

```

#Region "Variables"
    Protected DmCodigo As Integer
    Protected DmNombre As String
    Protected DmSqlCon As SqlConnection
    Protected DmEncontrar As Boolean
#End Region

#Region "Propertys"
    Public Property CodigoDepto() As String
        Get
            Return DmCodigo
        End Get
        Set(ByVal Value As String)
            DmCodigo = Value
        End Set
    End Property

    Public Property NombreDepartamento() As String
        Get
            Return DmNombre
        End Get
        Set(ByVal Value As String)
            DmNombre = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)
            DmSqlCon = Value
        End Set
    End Property
#End Region

#Region "Procedimientos"
    Public Function Lista() As DataSet
        Dim adp As New SqlDataAdapter("MODPER.SP_VERDEPARTAMENTOS", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function GuardarDepartamento() As Boolean
        Dim cmd As New SqlCommand("MODPER.SP_INSERTARDEPARTAMENTO", DmSqlCon)
        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
        Try
            cmd.CommandType = CommandType.StoredProcedure

```

```

        GuardarDepartamento = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ActualizarDepartamento() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_MODIFICARDEPARTAMENTO", DmSqlCon)
    cmd.Parameters.Add("@cod", SqlDbType.Int).Value = CodigoDepto
    cmd.Parameters.Add("@name", SqlDbType.VarChar, 250).Value = DmNombre
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarDepartamento = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIEmpleado: Encarga de comunicarse con la tabla de empleados dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIEmpleado
#Region "Variables"
    Protected DmCodigoEmp As Integer
    Protected DmDepto As String
    Protected DmNombres As String
    Protected DmApellidos As String
    Protected DmDireccion As String
    Protected DmTelefono As String
    Protected DmDui As String
    Protected DmNit As String
    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Property"
    Public Property CodigoEmpl() As String
        Get
            Return DmCodigoEmp
        End Get
        Set(ByVal Value As String)
            DmCodigoEmp = Value
        End Set
    End Property

```

```

Public Property DeptoEmpleado() As String
    Get
        Return DmDepto
    End Get
    Set(ByVal Value As String)
        DmDepto = Value
    End Set
End Property

Public Property NombreEmpleado() As String
    Get
        Return DmNombres
    End Get
    Set(ByVal Value As String)
        DmNombres = Value
    End Set
End Property

Public Property ApellidoEmpleado() As String
    Get
        Return DmApellidos
    End Get
    Set(ByVal Value As String)
        DmApellidos = Value
    End Set
End Property

Public Property DireccionEmpleado() As String
    Get
        Return DmDireccion
    End Get
    Set(ByVal Value As String)
        DmDireccion = Value
    End Set
End Property

Public Property TelefonoEmpleado() As String
    Get
        Return DmTelefono
    End Get
    Set(ByVal Value As String)
        DmTelefono = Value
    End Set
End Property

Public Property DuiEmpleado() As String
    Get
        Return DmDui
    End Get
    Set(ByVal Value As String)
        DmDui = Value
    End Set
End Property

Public Property NitEmpleado() As String
    Get

```

```

        Return DmNit
    End Get
    Set(ByVal Value As String)
        DmNit = Value
    End Set
End Property

Public Property EstadoEmpleado() As Boolean
    Get
        Return DmEstado
    End Get
    Set(ByVal Value As Boolean)
        DmEstado = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Procedimientos"
Public Function MostrarEmpleados() As DataSet ' Muestra los nombres y
apellidos de los empleados en base a su estado
    Dim adp As New SqlDataAdapter("MODPER.SP_VEREMPLEADOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function Lista() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_MOSTRAREMPLEADOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try

```

```

    Return ds
End Function

Public Function ObtenerDptos() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_MOSTRARDEPARTAMENTOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function EmpleadoActividad() As Boolean ' Comprueba si el empleado
tiene actividades pendientes activas asignadas
    Dim cmd As New SqlCommand("MODAGA.SP_EMPLEADOACTIVIDAD", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoEmp
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function GuardarEmpleado() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_INSERTAREMPLEADO", DmSqlCon)
    cmd.Parameters.Add("@NombreDepto", SqlDbType.VarChar, 250).Value = DmDepto
    cmd.Parameters.Add("@Nom", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Ape", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@Direcc", SqlDbType.VarChar, 350).Value = DmDireccion
    cmd.Parameters.Add("@Tel", SqlDbType.VarChar, 9).Value = DmTelefono
    cmd.Parameters.Add("@Dui", SqlDbType.VarChar, 10).Value = DmDui
    cmd.Parameters.Add("@Nit", SqlDbType.VarChar, 17).Value = DmNit
    cmd.Parameters.Add("@Estad", SqlDbType.Bit).Value = DmEstado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarEmpleado = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

```

```

    End Try
End Function

Public Function ActualizarEmpleado() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_MODIFICAREMPLLEADO", DmSqlCon)
    cmd.Parameters.Add("@codigo", SqlDbType.Int).Value = CodigoEmpl
    cmd.Parameters.Add("@NombreDepto", SqlDbType.VarChar, 250).Value =
DeptoEmpleado
    cmd.Parameters.Add("@name", SqlDbType.VarChar, 250).Value = NombreEmpleado
    cmd.Parameters.Add("@ape", SqlDbType.VarChar, 250).Value = ApellidoEmpleado
    cmd.Parameters.Add("@address", SqlDbType.VarChar, 250).Value =
DireccionEmpleado
    cmd.Parameters.Add("@tel", SqlDbType.VarChar, 9).Value = TelefonoEmpleado
    cmd.Parameters.Add("@du", SqlDbType.VarChar, 10).Value = DuiEmpleado
    cmd.Parameters.Add("@ni", SqlDbType.VarChar, 17).Value = NitEmpleado
    cmd.Parameters.Add("@estad", SqlDbType.Bit).Value = EstadoEmpleado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarEmpleado = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIHorario: Encarga de comunicarse con la tabla de horarios dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIHorario
#Region "Variables"
    Protected DmLlabe As Integer
    Protected DmCodEmpleado As Integer
    Protected DmDia As String
    Protected DmHoras As String
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertys"
    Public Property llabeHorario() As String
        Get
            Return DmLlabe
        End Get
        Set(ByVal Value As String)
            DmLlabe = Value
        End Set
    End Property

    Public Property HorarioCodEmpleado() As Integer
        Get

```



```

        Return DmCodEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodEmpleado = Value
    End Set
End Property

Public Property HorarioDia() As String
    Get
        Return DmDia
    End Get
    Set(ByVal Value As String)
        DmDia = Value
    End Set
End Property

Public Property HorarioHoras() As String
    Get
        Return DmHoras
    End Get
    Set(ByVal Value As String)
        DmHoras = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Procedimientos"
    Public Function MostrarEmpleados() As DataSet ' Muestra los nombres y
    apellidos de los empleados en base a su estado
        Dim adp As New SqlDataAdapter("MODPER.SP_VEREMPLEADOS", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value = True
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function Lista() As DataSet
        Dim adp As New SqlDataAdapter("MODPER.SP_VISUALIZAREMPLEADOS", DmSqlCon)
        Dim ds As New DataSet
        Try

```

```

        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function Lista2() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_VERHORARIOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodEmpleado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function GuardarHorario() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_INSERTARHORARIO", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.VarChar, 250).Value = DmCodEmpleado
    cmd.Parameters.Add("@Day", SqlDbType.VarChar, 250).Value = DmDia
    cmd.Parameters.Add("@Hour", SqlDbType.VarChar, 350).Value = DmHoras
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarHorario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ActualizarHorario() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_MODIFICARHORARIO", DmSqlCon)
    cmd.Parameters.Add("@llabe", SqlDbType.Int).Value = llabeHorario
    cmd.Parameters.Add("@CodEmp", SqlDbType.Int).Value = HorarioCodEmpleado
    cmd.Parameters.Add("@Day", SqlDbType.VarChar, 50).Value = HorarioDia
    cmd.Parameters.Add("@Hour", SqlDbType.VarChar, 4).Value = HorarioHoras
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarHorario = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()

```

```

        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIVacaciones: Encarga de comunicarse con la tabla de vacaciones dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIVacaciones
#Region "Variables"
    Protected DmCodVaca As Integer
    Protected DmCod As Integer
    Protected DmFechaInicio As Date
    Protected DmFechaFinal As Date
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertyys"
    Public Property CodigoVacacion() As Integer
    Get
        Return DmCodVaca
    End Get
    Set(ByVal Value As Integer)
        DmCodVaca = Value
    End Set
End Property

    Public Property CodigoEmpleado() As Integer
    Get
        Return DmCod
    End Get
    Set(ByVal Value As Integer)
        DmCod = Value
    End Set
End Property

    Public Property FechaInicio() As Date
    Get
        Return DmFechaInicio
    End Get
    Set(ByVal Value As Date)
        DmFechaInicio = Value
    End Set
End Property

    Public Property FechaFinal() As Date
    Get
        Return DmFechaFinal
    End Get
    Set(ByVal Value As Date)
        DmFechaFinal = Value
    End Set

```

```

End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Procedimientos"
Public Function MostrarEmpleados() As DataSet ' Muestra los nombres y
apellidos de los empleados en base a su estado
    Dim adp As New SqlDataAdapter("MODPER.SP_VEREMPLEADOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value = True
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function Lista() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_VISUALIZAREMPLEADOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function Lista2() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_VISUALIZARVACACIONES", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCod
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()

```

```

    End Try
    Return ds
End Function

Public Function GuardarVacaciones() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_INSERTARVACACION", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.VarChar, 250).Value = DmCod
    cmd.Parameters.Add("@date1", SqlDbType.Date).Value = DmFechaInicio.Date()
    cmd.Parameters.Add("@date2", SqlDbType.Date).Value = DmFechaFinal.Date()
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarVacaciones = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ActualizarVacaciones() As Boolean

    Dim cmd As New SqlCommand("MODPER.SP_MODIFICARVACACION", DmSqlCon)
    cmd.Parameters.Add("@llabe", SqlDbType.Int).Value = DmCodVaca
    cmd.Parameters.Add("@CodEmp", SqlDbType.Int).Value = DmCod
    cmd.Parameters.Add("@FechaInicio", SqlDbType.Date).Value = FechaInicio
    cmd.Parameters.Add("@FechaFinal", SqlDbType.Date).Value = FechaFinal
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarVacaciones = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIContratistas: Encarga de comunicarse con la tabla de contratistas dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIContratistas
#Region "Variables"
    Protected DmCodigo As Integer
    Protected DmNombres As String
    Protected DmApellidos As String
    Protected DmDireccion As String
    Protected DmTelefono As String
    Protected DmDui As String
    Protected DmNit As String

```

```

    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertys"
    Public Property CodigoContratista() As String
        Get
            Return DmCodigo
        End Get
        Set(ByVal Value As String)
            DmCodigo = Value
        End Set
    End Property

    Public Property NombreContratista() As String
        Get
            Return DmNombres
        End Get
        Set(ByVal Value As String)
            DmNombres = Value
        End Set
    End Property

    Public Property ApellidoContratista() As String
        Get
            Return DmApellidos
        End Get
        Set(ByVal Value As String)
            DmApellidos = Value
        End Set
    End Property

    Public Property DireccionContratista() As String
        Get
            Return DmDireccion
        End Get
        Set(ByVal Value As String)
            DmDireccion = Value
        End Set
    End Property

    Public Property TelefonoContratista() As String
        Get
            Return DmTelefono
        End Get
        Set(ByVal Value As String)
            DmTelefono = Value
        End Set
    End Property

    Public Property DuiContratista() As String
        Get
            Return DmDui
        End Get
        Set(ByVal Value As String)
            DmDui = Value
        End Set
    End Property

```

```

End Property

Public Property NitContratista() As String
    Get
        Return DmNit
    End Get
    Set(ByVal Value As String)
        DmNit = Value
    End Set
End Property

Public Property EstadoContratista() As Boolean
    Get
        Return DmEstado
    End Get
    Set(ByVal Value As Boolean)
        DmEstado = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Procedimientos"
Public Function Lista() As DataSet
    Dim adp As New SqlDataAdapter("MODPER.SP_VERCONTRATISTAS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function GuardarContratista() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_INSERTARCONTRATISTA", DmSqlCon)
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 250).Value = DmNombres
    cmd.Parameters.Add("@Apellidos", SqlDbType.VarChar, 250).Value = DmApellidos
    cmd.Parameters.Add("@Direcc", SqlDbType.VarChar, 350).Value = DmDireccion
    cmd.Parameters.Add("@Tel", SqlDbType.VarChar, 9).Value = DmTelefono
    cmd.Parameters.Add("@Du", SqlDbType.VarChar, 10).Value = DmDui
    cmd.Parameters.Add("@Ni", SqlDbType.VarChar, 17).Value = DmNit
    cmd.Parameters.Add("@Estad", SqlDbType.Bit).Value = DmEstado
    Try
        cmd.CommandType = CommandType.StoredProcedure

```

```

        GuardarContratista = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ActualizarContratista() As Boolean
    Dim cmd As New SqlCommand("MODPER.SP_MODIFICARCONTRATISTA", DmSqlCon)
    cmd.Parameters.Add("@codido", SqlDbType.Int).Value = CodigoContratista
    cmd.Parameters.Add("@name", SqlDbType.VarChar, 250).Value =
NombreContratista
    cmd.Parameters.Add("@ape", SqlDbType.VarChar, 250).Value =
ApellidoContratista
    cmd.Parameters.Add("@address", SqlDbType.VarChar, 250).Value =
DireccionContratista
    cmd.Parameters.Add("@tel", SqlDbType.VarChar, 9).Value = TelefonoContratista
    cmd.Parameters.Add("@du", SqlDbType.VarChar, 10).Value = DuiContratista
    cmd.Parameters.Add("@ni", SqlDbType.VarChar, 17).Value = NitContratista
    cmd.Parameters.Add("@estad", SqlDbType.Bit).Value = EstadoContratista
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarContratista = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CLASES DEL MODULO DE COBROS Y COSTOS (MODCYC)

ClIsam: Encarga de comunicarse con la tabla de contratistas dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class ClIsam
#Region "Variables"
    'Variables para guardar la ISAM
    Protected NumIsamVar As Integer
    Protected IsdemVar As Integer
    Protected SerieVar As String
    Protected NombreClienteVar As String
    Protected FechaVar As Date
    Protected EstadoVar As Boolean
    Protected TotalVar As Double

    'Variables para guardar en el DetalleIsam
    Protected CodIsamVar As Integer
    Protected NombreImpuestoVar As String

```



```

Protected NombreServicioVar As String
Protected SubTotalVar As Double

Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertys"
'Propiedades para capturar los datos de la ISAM
Public Property ProNumIsam() As String
    Get
        Return NumIsamVar
    End Get
    Set(ByVal Value As String)
        NumIsamVar = Value
    End Set
End Property

Public Property ProIsdem() As String
    Get
        Return IsdemVar
    End Get
    Set(ByVal Value As String)
        IsdemVar = Value
    End Set
End Property

Public Property ProSerie() As String
    Get
        Return SerieVar
    End Get
    Set(ByVal Value As String)
        SerieVar = Value
    End Set
End Property

Public Property ProNombreCliente() As String
    Get
        Return NombreClienteVar
    End Get
    Set(ByVal Value As String)
        NombreClienteVar = Value
    End Set
End Property

Public Property ProFecha() As String
    Get
        Return FechaVar
    End Get
    Set(ByVal Value As String)
        FechaVar = Value
    End Set
End Property

Public Property ProEstado() As Boolean
    Get
        Return EstadoVar
    End Get

```

```

        Set(ByVal Value As Boolean)
            EstadoVar = Value
        End Set
    End Property

    Public Property ProTotal() As String
        Get
            Return TotalVar
        End Get
        Set(ByVal Value As String)
            TotalVar = Value
        End Set
    End Property

'-----Propiedades para capturar los datos del DetalleIsam-----
    Public Property ProCodigoIsam() As String
        Get
            Return CodIsamVar
        End Get
        Set(ByVal Value As String)
            CodIsamVar = Value
        End Set
    End Property

    Public Property ProNombreImp() As String
        Get
            Return NombreImpuestoVar
        End Get
        Set(ByVal Value As String)
            NombreImpuestoVar = Value
        End Set
    End Property

    Public Property ProNombreSer() As String
        Get
            Return NombreServicioVar
        End Get
        Set(ByVal Value As String)
            NombreServicioVar = Value
        End Set
    End Property

    Public Property ProTotalDetalleIsam() As String
        Get
            Return TotalVar
        End Get
        Set(ByVal Value As String)
            TotalVar = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)

```

```

        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Funciones de la clase"
'Lista todas las isam para su seleccion
Public Function ListarIsam() As DataSet
    Dim adp As New SqlDataAdapter("MODRIN.SP_LISTARISAM", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
    End Try
    Return ds
End Function

'Muestra la tabla servicio en el combobox
Public Function ListaServicio() As DataSet
    Dim adp As New SqlDataAdapter("MODCYC.SP_MOSTRARNOMBRESERVICIO", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

'Muestra la tabla impuesto en el combobox
Public Function ListaImpuesto() As DataSet
    Dim adp As New SqlDataAdapter("MODCYC.SP_MOSTRARNOMBREIMPUESTO", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function MostrarDetallesISAM() As DataSet ' Muestra los servicios e
impuestos por ISAM
    Dim adp As New SqlDataAdapter("MODCYC.SP_VERIMPSE", DmSqlCon)
    Dim ds As New DataSet

```

```

    Try
        NumIsamVar adp.SelectCommand.Parameters.Add("@ISAM", SqlDbType.Int).Value =
        IsdemVar adp.SelectCommand.Parameters.Add("@ISDEM", SqlDbType.Int).Value =
        SerieVar adp.SelectCommand.Parameters.Add("@Serie", SqlDbType.VarChar, 2).Value =
        adp.SelectCommand.Parameters.Add("@NombreCliente", SqlDbType.VarChar,
250).Value = NombreClienteVar
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function ModificarTotal() As Boolean ' Modifica el total de la ISAM
    Dim cmd As New SqlCommand("MODCYC.SP_MODIFICARTOTAL", DmSqlCon)
    cmd.Parameters.Add("@Cliente", SqlDbType.VarChar, 250).Value =
ProNombreCliente
    cmd.Parameters.Add("@Estado", SqlDbType.Bit).Value = EstadoVar
    cmd.Parameters.Add("@Total", SqlDbType.Float).Value = TotalVar
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarTotal = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

'Funcion para agregar una ISAM
Public Function GuardarISAM() As Boolean
    Dim cmd As New SqlCommand("MODCYC.SP_INSERTARISAM", DmSqlCon)
    cmd.Parameters.Add("@NumIsa", SqlDbType.Int).Value = NumIsamVar
    cmd.Parameters.Add("@Isdemm", SqlDbType.Int).Value = IsdemVar
    cmd.Parameters.Add("@Seriee", SqlDbType.VarChar, 2).Value = SerieVar
    cmd.Parameters.Add("@NombreClient", SqlDbType.VarChar, 250).Value =
ProNombreCliente
    cmd.Parameters.Add("@Datee", SqlDbType.Date).Value = FechaVar
    cmd.Parameters.Add("@Status", SqlDbType.Bit).Value = EstadoVar
    cmd.Parameters.Add("@Total1", SqlDbType.Float).Value = TotalVar
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarISAM = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try

```

```

End Function

Public Function GuardarDetalleIsam() As Boolean ' Guarda cada detalle de la
ISAM
    Dim cmd As New SqlCommand("MODCYC.SP_INSERTARDETALLEISAM", DmSqlCon)
    cmd.Parameters.Add("@NombreServicio", SqlDbType.VarChar, 250).Value =
ProNombreSer
    cmd.Parameters.Add("@NombreImpuesto", SqlDbType.VarChar, 250).Value =
ProNombreImp
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarDetalleIsam = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

ClServicio: Encarga de comunicarse con la tabla de servicios dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class ClServicios
#Region "Variables"
    Property DmCodigoServicio As Integer
    Protected DmNombre As String
    Protected DmDescripcion As String
    Protected DmValor As Double
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertys"
    Public Property CodigoServicio() As String
    Get
        Return DmCodigoServicio
    End Get
    Set(ByVal Value As String)
        DmCodigoServicio = Value
    End Set
End Property

    Public Property NombreServicio() As String
    Get
        Return DmNombre
    End Get
    Set(ByVal Value As String)
        DmNombre = Value
    End Set
End Property

```

```

Public Property DescripcionServicio() As String
    Get
        Return DmDescripcion
    End Get
    Set(ByVal Value As String)
        DmDescripcion = Value
    End Set
End Property

Public Property ValorServicio() As String
    Get
        Return DmValor
    End Get
    Set(ByVal Value As String)
        DmValor = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Procedimientos"
Public Function Lista() As DataSet
    Dim adp As New SqlDataAdapter("MODCYC.SP_VERSERVICIOS", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function GuardarServicio() As Boolean
    Dim cmd As New SqlCommand("MODCYC.SP_INSERTARSERVICIO", DmSqlCon)
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 250).Value = DmNombre
    cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 350).Value =
DmDescripcion
    cmd.Parameters.Add("@Valor", SqlDbType.Float).Value = DmValor
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarServicio = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally

```

```

        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
Public Function ActualizarServicio() As Boolean
    Dim cmd As New SqlCommand("MODCYC.SP_MODIFICARSERVICIO", DmSqlCon)
    cmd.Parameters.Add("@CodServicio", SqlDbType.Int).Value = CodigoServicio
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 250).Value = NombreServicio
    cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 350).Value =
DescripcionServicio
    cmd.Parameters.Add("@Valor", SqlDbType.Float).Value = ValorServicio
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarServicio = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CImpuesto: Encarga de comunicarse con la tabla de impuestos dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CImpuesto
#Region "Variables"
    Property DmCodigoImpuesto As Integer
    Protected DmNombre As String
    Protected DmDescripcion As String
    Protected DmValor As Double
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propertys"
    Public Property CodigoImpuesto() As String
    Get
        Return DmCodigoImpuesto
    End Get
    Set(ByVal Value As String)
        DmCodigoImpuesto = Value
    End Set
    End Property

    Public Property NombreImpuesto() As String
    Get
        Return DmNombre
    End Get
    Set(ByVal Value As String)
        DmNombre = Value
    End Set
    End Property

```

```

        End Set
    End Property

    Public Property DescripcionImpuesto() As String
        Get
            Return DmDescripcion
        End Get
        Set(ByVal Value As String)
            DmDescripcion = Value
        End Set
    End Property

    Public Property ValorImpuesto() As String
        Get
            Return DmValor
        End Get
        Set(ByVal Value As String)
            DmValor = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)
            DmSqlCon = Value
        End Set
    End Property
#End Region

#Region "Procedimientos"
    Public Function Lista() As DataSet
        Dim adp As New SqlDataAdapter("MODCYC.SP_VERIMPUESTOS", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function GuardarImpuesto() As Boolean
        Dim cmd As New SqlCommand("MODCYC.SP_INSERTARIMPUESTO", DmSqlCon)
        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 250).Value = DmNombre
        cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 350).Value =
DmDescripcion
        cmd.Parameters.Add("@Valor", SqlDbType.Float).Value = DmValor
        Try
            cmd.CommandType = CommandType.StoredProcedure
            GuardarImpuesto = cmd.ExecuteNonQuery()
        Catch

```



```

        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
Public Function ActualizarImpuesto() As Boolean
    Dim cmd As New SqlCommand("MODCYC.SP_MODIFICARIMPUESTO", DmSqlCon)
    cmd.Parameters.Add("@CodImpuesto", SqlDbType.Int).Value = CodigoImpuesto
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 250).Value = NombreImpuesto
    cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 350).Value =
DescripcionImpuesto
    cmd.Parameters.Add("@Valor", SqlDbType.Float).Value = ValorImpuesto
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ActualizarImpuesto = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIBuscarCliente: Permite realizar operaciones de búsqueda con respecto a los clientes.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient

Public Class CIBuscarCliente
    #Region "Variables"
        Protected DmClient As String
        Protected DmSqlCon As SqlConnection
        Protected DmEncontrar As Boolean
        Private DmNombre As String
    #End Region

    #Region "Property's"
        Public Property Cliente() As String
            Get
                Return DmClient
            End Get
            Set(ByVal Value As String)
                DmClient = Value
            End Set
        End Property

        Public Property GetConexion() As SqlConnection
            Get
                Return DmSqlCon
            End Get
            Set(ByVal Value As SqlConnection)
                DmSqlCon = Value
            End Set
        End Property
    End Class

```

```

        End Set
    End Property

    Public Property Buscar() As Boolean
        Get
            Return DmEncontrar
        End Get
        Set(ByVal value As Boolean)
            DmEncontrar = value
        End Set
    End Property

    Public Property NombreCliente() As String
        Get
            Return DmNombre
        End Get
        Set(ByVal Value As String)
            DmNombre = Value
        End Set
    End Property
#End Region

#Region "Procedimiento"
    Public Function Lista() As DataSet
        Dim adp As New SqlDataAdapter("MODCYC.SP_VERCLIENTES", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function
#End Region
End Class

```

CLASES DEL MODULO DE AGENDA (MODAGA)

CIPlan: Encargada de comunicarse con la tabla de los planes de trabajo dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIPlan
#Region "Campos"
    Protected DmCodigoPlan As Integer
    Protected DmNombre As String
    Protected DmDescripcion As String
    Protected DmAnio As Integer
    Protected DmSqlCon As SqlConnection

```

```

#End Region

#Region "Propiedades"
' Propertys de los campos de la tabla de los planes de trabajo
Public Property CodigoPlan() As Integer ' Codigo del plan de trabajo
    Get
        Return DmCodigoPlan
    End Get
    Set(ByVal Value As Integer)
        DmCodigoPlan = Value
    End Set
End Property

Public Property NombrePlan() As String ' Nombre del plan de trabajo
    Get
        Return DmNombre
    End Get
    Set(ByVal Value As String)
        DmNombre = Value
    End Set
End Property

Public Property DescripcionPlan() As String ' Descripcion del plan de trabajo
    Get
        Return DmDescripcion
    End Get
    Set(ByVal Value As String)
        DmDescripcion = Value
    End Set
End Property

Public Property AnioPlan As Integer ' Año del plan de trabajo
    Get
        Return DmAnio
    End Get
    Set(ByVal value As Integer)
        DmAnio = value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Métodos"
Public Function MostrarPlanes() As DataSet ' Muestra los planes de trabajo
    Dim adp As New SqlDataAdapter("MODAGA.SP_VERPLANES", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    End Try
End Function

```

```

        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function GuardarPlan() As Boolean ' Guarda un nuevo plan de trabajo
        Dim cmd As New SqlCommand("MODAGA.SP_GUARDARPLAN", DmSqlCon)
        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
        cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 150).Value =
DmDescripcion
        cmd.Parameters.Add("@Anio", SqlDbType.Int).Value = DmAnio
        Try
            cmd.CommandType = CommandType.StoredProcedure
            GuardarPlan = cmd.ExecuteNonQuery()
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function

    Public Function ModificarPlan() As Boolean ' Modifica el plan de trabajo en base
a su codigo
        Dim cmd As New SqlCommand("MODAGA.SP_MODIFICARPLAN", DmSqlCon)
        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoPlan
        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
        cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 150).Value =
DmDescripcion
        cmd.Parameters.Add("@Anio", SqlDbType.Int).Value = DmAnio
        Try
            cmd.CommandType = CommandType.StoredProcedure
            ModificarPlan = cmd.ExecuteNonQuery()
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function

    Public Function VerificarPlan() As Boolean ' Verifica si un plan esta siendo
ingresado dos veces
        Dim cmd As New SqlCommand("MODAGA.SP_VERIFICARPLAN", DmSqlCon)
        Dim data As SqlDataReader
        Try
            cmd.CommandType = CommandType.StoredProcedure
            cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
            data = cmd.ExecuteReader()
            If data.Read Then
                Return True
            Else
                Return False
            End If
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function

```

```

        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Close()
    End Try
End Function
#End Region
End Class

```

CIActividad: Encargada de comunicarse con la tabla de actividades dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIActividad
#Region "Campos"
    Protected DmCodigoAct As Integer
    Protected DmNombre As String
    Protected DmDescripcion As String
    Protected DmMotivo As String
    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
    ' Propertys de los campos de la tabla de actividades
    Public Property CodigoAct() As Integer ' Codigo de la actividad
        Get
            Return DmCodigoAct
        End Get
        Set(ByVal Value As Integer)
            DmCodigoAct = Value
        End Set
    End Property

    Public Property NombreAct() As String ' Nombre de la actividad
        Get
            Return DmNombre
        End Get
        Set(ByVal Value As String)
            DmNombre = Value
        End Set
    End Property

    Public Property DescripcionAct() As String ' Descripcion de la actividad
        Get
            Return DmDescripcion
        End Get
        Set(ByVal Value As String)
            DmDescripcion = Value
        End Set
    End Property

```

```

    Public Property Motivo() As String ' Motivo de modificacion/eliminacion de la
actividad
    Get
        Return DmMotivo
    End Get
    Set(ByVal Value As String)
        DmMotivo = Value
    End Set
End Property

Public Property Estado() As Boolean ' Estado de la actividad
    Get
        Return DmEstado
    End Get
    Set(ByVal Value As Boolean)
        DmEstado = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Métodos"
    Public Function MostrarActividades() As DataSet ' Muestra las actividades en
base a su estado
    Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADES", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

    Public Function GuardarActividad() As Boolean ' Guarda una nueva actividad
    Dim cmd As New SqlCommand("MODAGA.SP_GUARDARACTIVIDAD", DmSqlCon)
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
    cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 150).Value =
DmDescripcion
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarActividad = cmd.ExecuteNonQuery()
    Catch

```

```

        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ModificarActividad() As Boolean ' Modifica una actividad en base
a su codigo
    Dim cmd As New SqlCommand("MODAGA.SP_MODIFICARACTIVIDAD", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoAct
    cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
    cmd.Parameters.Add("@Descripcion", SqlDbType.VarChar, 150).Value =
DmDescripcion
    cmd.Parameters.Add("@Motivo", SqlDbType.VarChar, 250).Value = DmMotivo
    cmd.Parameters.Add("@Estado", SqlDbType.Bit).Value = DmEstado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarActividad = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function VerificarActividad() As Boolean ' Verifica si una actividad
esta siendo ingresada dos veces
    Dim cmd As New SqlCommand("MODAGA.SP_VERIFICARACTIVIDAD", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar, 50).Value = DmNombre
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Close()
    End Try
End Function

Public Function ActividadEmpleado() As Boolean ' Comprueba si una actividad
pendiente y activa ya ha sido asignada a uno o mas empleados
    Dim cmd As New SqlCommand("MODAGA.SP_ACTIVIDADEMPLEADO", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoAct
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then

```

```

        Return True
    Else
        Return False
    End If
Catch
    Throw
Finally
    cmd.Dispose()
    DmSqlCon.Dispose()
End Try
End Function

Public Function ActividadEmpleado(ByVal Anio As Integer) As Boolean '
Comprueba si una actividad pendiente y activa ya ha sido asignada a uno o mas
empleados
    Dim cmd As New SqlCommand("MODAGA.SP_ACTIVIDADEMPLEADOANIO", DmSqlCon) '
dependiendo de año de plan
    Dim data As SqlDataReader
    Try
        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoAct
        cmd.Parameters.Add("@Inicio", SqlDbType.Date).Value = New Date(Anio, 1,
1)
        cmd.Parameters.Add("@Fin", SqlDbType.Date).Value = New Date(Anio, 12,
31)
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function EmpleadoActividad() As Boolean ' Comprueba si el empleado
tiene actividades pendientes activas asignadas
    Dim cmd As New SqlCommand("MODAGA.SP_EMPLEADOACTIVIDAD", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoAct
        cmd.CommandType = CommandType.StoredProcedure
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()

```



```

        End Try
    End Function

    Public Function EmpleadoActividad(ByVal Anio As Integer) As Boolean '
    Comprueba si el empleado tiene actividades pendientes activas asignadas
    dependiendo del año
        Dim cmd As New SqlCommand("MODAGA.SP_EMPLEADOACTIVIDADANIO", DmSqlCon) '
        Dim data As SqlDataReader
        Try
            cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoAct
            cmd.Parameters.Add("@Inicio", SqlDbType.Date).Value = New Date(Anio, 1,
1)
            cmd.Parameters.Add("@Fin", SqlDbType.Date).Value = New Date(Anio, 12,
31)
            cmd.CommandType = CommandType.StoredProcedure
            data = cmd.ExecuteReader()
            If data.Read Then
                Return True
            Else
                Return False
            End If
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function
#End Region
End Class

```

CIAP: Encargada de comunicarse con la tabla que relaciona a las actividades con los planes de trabajo dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIAP
#Region "Campos"
    Protected DmCodigoPlan As Integer
    Protected DmCodigoActividad As Integer
    Protected DmMotivo As String
    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
' Propertys de la clase
    Public Property CodigoPlan() As Integer ' Codigo del plan de trabajo
    Get
        Return DmCodigoPlan
    End Get
    Set(ByVal Value As Integer)
        DmCodigoPlan = Value
    End Set
End Property

```

```

Public Property CodigoActividad() As Integer 'Codigo de la actividad
    Get
        Return DmCodigoActividad
    End Get
    Set(ByVal Value As Integer)
        DmCodigoActividad = Value
    End Set
End Property

Public Property Motivo() As String 'Motivo de la restauracion/eliminacion de
la actividad
    Get
        Return DmMotivo
    End Get
    Set(ByVal Value As String)
        DmMotivo = Value
    End Set
End Property

Public Property EstadoActividad() As Boolean 'Estado de la actividad
    Get
        Return DmEstado
    End Get
    Set(ByVal Value As Boolean)
        DmEstado = Value
    End Set
End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Métodos"
Public Function MostrarActividadesPorPlan() As DataSet 'Muestra las
actividades en base a codigo de plan y estado de la actividad
    Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESPLAN", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoPlan
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try

```

```

        Return ds
    End Function

    Public Function MostrarActividadesPorPlanTodo() As DataSet ' Muestra las
    actividades en base a codigo de plan
        Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESPLANTODO", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoPlan
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function GuardarActividadAlPlan() As Boolean ' Agrega una nueva
    actividad al plan de trabajo
        Dim cmd As New SqlCommand("MODAGA.SP_GUARDARACTIVIDADPLAN", DmSqlCon)
        cmd.Parameters.Add("@CodigoPlan", SqlDbType.Int).Value = DmCodigoPlan
        cmd.Parameters.Add("@CodigoActividad", SqlDbType.Int).Value =
DmCodigoActividad
        Try
            cmd.CommandType = CommandType.StoredProcedure
            GuardarActividadAlPlan = cmd.ExecuteNonQuery()
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function

    Public Function ModificarActividadDelPlan() As Boolean ' Modifica una actividad
    del plan de trabajo
        Dim cmd As New SqlCommand("MODAGA.SP_MODIFICARACTIVIDADPLAN", DmSqlCon)
        cmd.Parameters.Add("@CodigoPlan", SqlDbType.Int).Value = DmCodigoPlan
        cmd.Parameters.Add("@CodigoActividad", SqlDbType.Int).Value =
DmCodigoActividad
        cmd.Parameters.Add("@Motivo", SqlDbType.VarChar, 250).Value = DmMotivo
        cmd.Parameters.Add("@Estado", SqlDbType.Bit).Value = DmEstado
        Try
            cmd.CommandType = CommandType.StoredProcedure
            ModificarActividadDelPlan = cmd.ExecuteNonQuery()
        Catch
            Throw
        Finally
            cmd.Dispose()
            DmSqlCon.Dispose()
        End Try
    End Function

```

```

    End Function
#End Region
End Class

```

CIEP: Encargada de comunicarse con la tabla que relaciona a los empleados con los planes de trabajo dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIEP
#Region "Campos"
    Protected DmCodigoPlan As Integer
    Protected DmCodigoEmpleado As Integer
    Protected DmMotivo As String
    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
' Propertys de los campos
Public Property CodigoPlan() As Integer ' Codigo del plan de trabajo
    Get
        Return DmCodigoPlan
    End Get
    Set(ByVal Value As Integer)
        DmCodigoPlan = Value
    End Set
End Property

Public Property CodigoEmpleado() As Integer ' Codigo del empleado
    Get
        Return DmCodigoEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodigoEmpleado = Value
    End Set
End Property

Public Property Motivo() As String ' Motivo de la restauracion/eliminacion del
empleado
    Get
        Return DmMotivo
    End Get
    Set(ByVal Value As String)
        DmMotivo = Value
    End Set
End Property

Public Property Estado() As Boolean ' Estado del empleado
    Get
        Return DmEstado
    End Get
    Set(ByVal Value As Boolean)
        DmEstado = Value
    End Set

```

```

End Property

Public Property GetConexion() As SqlConnection
    Get
        Return DmSqlCon
    End Get
    Set(ByVal Value As SqlConnection)
        DmSqlCon = Value
    End Set
End Property
#End Region

#Region "Métodos"
Public Function MostrarEmpleadosPorPlan() As DataSet ' Muestra los empleados en
base a codigo de plan y estado del empleado
    Dim adp As New SqlDataAdapter("MODAGA.SP_VEREMPLEADOSPLAN", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoPlan
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function MostrarEmpleadosPorPlanTodo() As DataSet ' Muestra los
empleados en base a codigo de plan
    Dim adp As New SqlDataAdapter("MODAGA.SP_VEREMPLEADOSPLANTODO", DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoPlan
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function GuardarEmpleadoAlPlan() As Boolean ' Agrega un nuevo empleado
al plan de trabajo
    Dim cmd As New SqlCommand("MODAGA.SP_GUARDAREMPLADOPLAN", DmSqlCon)
    cmd.Parameters.Add("@CodigoPlan", SqlDbType.Int).Value = DmCodigoPlan

```

```

        cmd.Parameters.Add("@CodigoEmpleado", SqlDbType.Int).Value =
DmCodigoEmpleado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarEmpleadoAlPlan = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function ModificarEmpleadoDelPlan() As Boolean ' Modifica un empleado de
plan de trabajo
Dim cmd As New SqlCommand("MODAGA.SP_MODIFICAREMPLEADOPLAN", DmSqlCon)
cmd.Parameters.Add("@CodigoPlan", SqlDbType.Int).Value = DmCodigoPlan
cmd.Parameters.Add("@CodigoEmpleado", SqlDbType.Int).Value =
DmCodigoEmpleado
cmd.Parameters.Add("@Motivo", SqlDbType.VarChar, 250).Value = DmMotivo
cmd.Parameters.Add("@Estado", SqlDbType.Bit).Value = DmEstado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarEmpleadoDelPlan = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

CIEA: Encargada de comunicarse con la tabla que relaciona a los empleados con sus actividades dentro de la base de datos.

```

Imports System
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient
Public Class CIEA
#Region "Campos"
    Protected DmCodigoEA As Integer
    Protected DmCodigoEmpleado As Integer
    Protected DmCodigoActividad As Integer
    Protected DmFechaInicio As Date
    Protected DmFechaFin As Date
    Protected DmFinalizado As Boolean
    Protected DmMotivo As String
    Protected DmEstado As Boolean
    Protected DmSqlCon As SqlConnection
#End Region

#Region "Propiedades"
' Propertys de los campos de la tabla de los planes de trabajo
Public Property CodigoEA() As Integer ' Codigo del empleado/actividad

```

```

    Get
        Return DmCodigoEA
    End Get
    Set(ByVal Value As Integer)
        DmCodigoEA = Value
    End Set
End Property

Public Property CodigoEmpleado() As Integer ' Codigo del empleado
    Get
        Return DmCodigoEmpleado
    End Get
    Set(ByVal Value As Integer)
        DmCodigoEmpleado = Value
    End Set
End Property

Public Property CodigoActividad() As Integer ' Codigo de la actividad
    Get
        Return DmCodigoActividad
    End Get
    Set(ByVal Value As Integer)
        DmCodigoActividad = Value
    End Set
End Property

Public Property FechaInicio() As Date ' Fecha de inicio de la actividad
    Get
        Return DmFechaInicio
    End Get
    Set(ByVal Value As Date)
        DmFechaInicio = Value
    End Set
End Property

Public Property FechaFin() As Date ' Fecha de fin de la actividad
    Get
        Return DmFechaFin
    End Get
    Set(ByVal Value As Date)
        DmFechaFin = Value
    End Set
End Property

Public Property Finalizado() As Boolean ' Indica si la actividad se ha
finalizado o no
    Get
        Return DmFinalizado
    End Get
    Set(ByVal Value As Boolean)
        DmFinalizado = Value
    End Set
End Property

Public Property Motivo() As String ' Motivo de la cancelacion de la actividad
    Get
        Return DmMotivo

```

```

        End Get
        Set(ByVal Value As String)
            DmMotivo = Value
        End Set
    End Property

    Public Property Estado() As Boolean ' Estado de la actividad
        Get
            Return DmEstado
        End Get
        Set(ByVal Value As Boolean)
            DmEstado = Value
        End Set
    End Property

    Public Property GetConexion() As SqlConnection
        Get
            Return DmSqlCon
        End Get
        Set(ByVal Value As SqlConnection)
            DmSqlCon = Value
        End Set
    End Property
#End Region

#Region "Métodos"
    Public Function MostrarActividadesDeEmpleado() As DataSet ' Muestra las
    actividades de los empleados basandose en su codigo
        Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESEMPLEADOS", DmSqlCon)
    ' y estado de la actividad y si ha finalizado o no
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoEmpleado
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
            adp.SelectCommand.Parameters.Add("@FechaInicio", SqlDbType.Date).Value =
DmFechaInicio
            adp.SelectCommand.Parameters.Add("@FechaFin", SqlDbType.Date).Value =
DmFechaFin
            adp.SelectCommand.Parameters.Add("@Finalizado", SqlDbType.Bit).Value =
DmFinalizado
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function MostrarActividadesDeEmpleadNoProgreso() As DataSet ' Muestra
    las actividades de los empleados que no
        Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESPENDIENTES",
DmSqlCon) ' estan en progreso

```



```

        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Codigo", SqlDbType.Int).Value =
DmCodigoEmpleado
            adp.SelectCommand.Parameters.Add("@FechaInicio", SqlDbType.Date).Value =
DmFechaInicio
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
            adp.SelectCommand.Parameters.Add("@Finalizado", SqlDbType.Bit).Value =
DmFinalizado
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function MostrarActividadesEnProgreso() As DataSet ' Muestra las
actividades que estan en progreso y no han sido canceladas
        Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESPROGRESO", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Fecha", SqlDbType.Date).Value =
DmFechaInicio
            adp.SelectCommand.Parameters.Add("@Finalizado", SqlDbType.Bit).Value =
DmFinalizado
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally
            adp.Dispose()
            DmSqlCon.Dispose()
        End Try
        Return ds
    End Function

    Public Function MostrarActividadesDelDia() As DataSet ' Muestra las actividades
del día que no han sido canceladas ni finalizadas
        Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADES DIA", DmSqlCon)
        Dim ds As New DataSet
        Try
            adp.SelectCommand.Parameters.Add("@Fecha", SqlDbType.Date).Value =
DmFechaInicio
            adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
            adp.SelectCommand.CommandType = CommandType.StoredProcedure
            adp.Fill(ds)
        Catch
            Throw
        Finally

```

```

        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function MostrarActividadesPendientes() As DataSet ' Muestra las
actividades no finalizadas entre dos fechas
    Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESPENDIENTESFECHAS",
DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@FechaInicio", SqlDbType.Date).Value =
DmFechaInicio
        adp.SelectCommand.Parameters.Add("@FechaFin", SqlDbType.Date).Value =
DmFechaFin
        adp.SelectCommand.Parameters.Add("@Finalizado", SqlDbType.Bit).Value =
DmFinalizado
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

Public Function MostrarActividadesRealizadas() As DataSet ' Muestra las
actividades finalizadas entre dos fechas
    Dim adp As New SqlDataAdapter("MODAGA.SP_VERACTIVIDADESFINALIZADAS",
DmSqlCon)
    Dim ds As New DataSet
    Try
        adp.SelectCommand.Parameters.Add("@FechaInicio", SqlDbType.Date).Value =
DmFechaInicio
        adp.SelectCommand.Parameters.Add("@FechaFin", SqlDbType.Date).Value =
DmFechaFin
        adp.SelectCommand.Parameters.Add("@Finalizado", SqlDbType.Bit).Value =
DmFinalizado
        adp.SelectCommand.Parameters.Add("@Estado", SqlDbType.Bit).Value =
DmEstado
        adp.SelectCommand.CommandType = CommandType.StoredProcedure
        adp.Fill(ds)
    Catch
        Throw
    Finally
        adp.Dispose()
        DmSqlCon.Dispose()
    End Try
    Return ds
End Function

```

```

    Public Function GuardarActividadEmpleado() As Boolean ' Agrega una actividad
al empleado
    Dim cmd As New SqlCommand("MODAGA.SP_GUARDARACTIVIDADEMPLEADO", DmSqlCon)
    cmd.Parameters.Add("@CodigoEmpleado", SqlDbType.Int).Value =
DmCodigoEmpleado
    cmd.Parameters.Add("@CodigoActividad", SqlDbType.Int).Value =
DmCodigoActividad
    cmd.Parameters.Add("@Inicio", SqlDbType.Date).Value = DmFechaInicio
    Try
        cmd.CommandType = CommandType.StoredProcedure
        GuardarActividadEmpleado = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

    Public Function ModificarFechaActividad() As Boolean ' Modifica la fecha de
inicio de actividad asignada al empleado
    Dim cmd As New SqlCommand("MODAGA.SP_MODIFICARFECHACTIVIDAD", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoEA
    cmd.Parameters.Add("@Fecha", SqlDbType.Date).Value = DmFechaInicio
    cmd.Parameters.Add("@Motivo", SqlDbType.VarChar, 250).Value = DmMotivo
    cmd.Parameters.Add("@Finalizado", SqlDbType.Bit).Value = DmFinalizado
    Try
        cmd.CommandType = CommandType.StoredProcedure
        ModificarFechaActividad = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

    Public Function VerificarActividadEmpleado() As Boolean ' Verifica si una
actividad esta siendo asignada dos veces
    Dim cmd As New SqlCommand("MODAGA.SP_VERIFICARACTIVIDADEMPLEADO", DmSqlCon)
    Dim data As SqlDataReader
    Try
        cmd.CommandType = CommandType.StoredProcedure
        cmd.Parameters.Add("@CodigoEmpleado", SqlDbType.Int).Value =
DmCodigoEmpleado
        cmd.Parameters.Add("@CodigoActividad", SqlDbType.Int).Value =
DmCodigoActividad
        cmd.Parameters.Add("@Fecha", SqlDbType.Date).Value = DmFechaInicio
        data = cmd.ExecuteReader()
        If data.Read Then
            Return True
        Else
            Return False
        End If
    Catch
        Throw
    Finally

```

```

        cmd.Dispose()
        DmSqlCon.Close()
    End Try
End Function

Public Function FinalizarActividad() As Boolean ' Finaliza la actividad en
progreso seleccionada
    Dim cmd As New SqlCommand("MODAGA.SP_FINALIZARACTIVIDAD", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoEA
    cmd.Parameters.Add("@Fecha", SqlDbType.Date).Value = DmFechaFin
    Try
        cmd.CommandType = CommandType.StoredProcedure
        FinalizarActividad = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function

Public Function CancelarActividad() As Boolean ' Cancela la actividad asignada
al empleado seleccionado
    Dim cmd As New SqlCommand("MODAGA.SP_CANCELARACTIVIDAD", DmSqlCon)
    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = DmCodigoEA
    cmd.Parameters.Add("@Motivo", SqlDbType.VarChar, 250).Value = DmMotivo
    Try
        cmd.CommandType = CommandType.StoredProcedure
        CancelarActividad = cmd.ExecuteNonQuery()
    Catch
        Throw
    Finally
        cmd.Dispose()
        DmSqlCon.Dispose()
    End Try
End Function
#End Region
End Class

```

DISPARADORES

Los disparadores son utilizados para ejecutar actividades dentro de la base de datos sin que el usuario pueda percibirlo y sin necesidad de realizar algún tipo de inserción de datos dentro del código fuente. Para el caso estos disparadores se han utilizado para la elaboración de “logs” transaccionales como seguridad del sistema.

```
--DISPARADOR [MODRIN].[TRGPARTIDADEFUNCION] QUE SE UTILIZA PARA REALIZAR
UN LOG TRANSACCIONAL DE LAS PARTIDAS DE DEFUNCION.
```

```
USE [DBCEMENTERIO]
GO
/***** Object: Trigger [MODRIN].[TRGPARTIDADEFUNCION]      Script Date:
04/06/2011 12:52:16 *****/
```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TRIGGER [MODRIN].[TRGPARTIDADEFUNCION] ON
[MODRIN].[TBLPARTIDADEFUNCION] FOR INSERT,UPDATE,DELETE AS

DECLARE @now datetime,@TipoTransaccion varchar(10)

BEGIN TRY

SET @now = getdate()

    if exists (select * from inserted)
        if exists (select * from deleted)
            select @TipoTransaccion = 'U'
        else
            select @TipoTransaccion = 'I'
    else
        select @TipoTransaccion = 'D'

    IF (@TipoTransaccion='U' or @TipoTransaccion='I')
        BEGIN
            INSERT INTO [MODRIN].[TBLPARTIDADEFUNCION_AUDITORIA]
            (NumeroPartidaDefuncion,CodIsam,NombreSolicitante,NombreFallecido,Edad,Es
tadoCivil,OrigenFallecido,DireccionFallecido,OficioProfesionFallecidos,No
mbrePadre,NombreMadre,MedicoAvalaDefuncion,HoraFechaFallecimiento,CausaMu
erte,CodEmpleado, Accion, FechaActualizacion)
            SELECT
            INSERTED.[NumeroPartidaDefuncion],INSERTED.CodIsam,INSERTED.NombreSolicit
ante,INSERTED.NombreFallecido,INSERTED.Edad,INSERTED.EstadoCivil,INSERTED
.OrigenFallecido,INSERTED.DireccionFallecido,INSERTED.OficioProfesionFall
ecidos,INSERTED.NombrePadre,INSERTED.NombreMadre,INSERTED.MedicoAvalaDefu
ncion,INSERTED.HoraFechaFallecimiento,INSERTED.CausaMuerte,INSERTED.CodEm
pleado ,@TipoTransaccion, @now
            FROM INSERTED
        END
    ELSE
        BEGIN
            INSERT INTO [MODRIN].[TBLPARTIDADEFUNCION_AUDITORIA]
            (NumeroPartidaDefuncion,CodIsam,NombreSolicitante,NombreFallecido,Edad,Es
tadoCivil,OrigenFallecido,DireccionFallecido,OficioProfesionFallecidos,No
mbrePadre,NombreMadre,MedicoAvalaDefuncion,HoraFechaFallecimiento,CausaMu
erte,CodEmpleado, Accion, FechaActualizacion)
            SELECT
            DELETED.[NumeroPartidaDefuncion],DELETED.CodIsam,DELETED.NombreSolicitant
e,DELETED.NombreFallecido,DELETED.Edad,DELETED.EstadoCivil,DELETED.Origen
Fallecido,DELETED.DireccionFallecido,DELETED.OficioProfesionFallecidos,DE
LETED.NombrePadre,DELETED.NombreMadre,DELETED.MedicoAvalaDefuncion,DELETE
D.HoraFechaFallecimiento,DELETED.CausaMuerte,DELETED.CodEmpleado
,@TipoTransaccion, @now
            FROM DELETED
        END
END TRY

BEGIN CATCH

```

```

ROLLBACK TRANSACTION
RAISERROR ('HUBO UN ERROR EN LA INSERCIÓN',20,1)
END CATCH

--DISPARADOR [MODRIN].[TRGTBLINHUMACION]QUE SE UTILIZA PARA REALIZAR UN
LOG TRANSACCIONAL DE LAS PARTIDAS DE INHUMACION.
USE [DBCementerio]
GO
/***** Object: Trigger [MODRIN].[TRGTBLINHUMACION]      Script Date:
04/06/2011 12:59:56 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TRIGGER [MODRIN].[TRGTBLINHUMACION] ON [MODRIN].[TBLINHUMACION]
FOR INSERT,UPDATE,DELETE AS

DECLARE @now datetime,@TipoTransaccion varchar(10)

BEGIN TRY

SET @now = getdate()

    if exists (select * from inserted)
        if exists (select * from deleted)
            select @TipoTransaccion = 'U'
        else
            select @TipoTransaccion = 'I'
    else
        select @TipoTransaccion = 'D'

    IF (@TipoTransaccion='U' or @TipoTransaccion='I')
        BEGIN
            INSERT INTO [MODRIN].[TBLINHUMACION_AUDITORIA]
(CodInhumacion,CodIsam,NumeroPartidaDefuncion,CodPropiedad,CodEmpleado,C
ausasFallecimiento,LugarFallecimiento,DireccionFallecimiento,FechaInhumac
ion,Accion,FechaActualizacion)
            SELECT
INSERTED.[CodInhumacion],INSERTED.[CodIsam],INSERTED.[NumeroPartidaDefunc
ion],INSERTED.[CodPropiedad],INSERTED.[CodEmpleado],INSERTED.[CausasFall
ecimiento],INSERTED.[LugarFallecimiento],INSERTED.[DireccionFallecimiento
],INSERTED.[FechaInhumacion], @TipoTransaccion, @now
            FROM INSERTED
        END
    ELSE
        BEGIN
            INSERT INTO [MODRIN].[TBLINHUMACION_AUDITORIA]
(CodInhumacion,CodIsam,NumeroPartidaDefuncion,CodPropiedad,CodEmpleado,C
ausasFallecimiento,LugarFallecimiento,DireccionFallecimiento,FechaInhumac
ion,Accion,FechaActualizacion)
            SELECT
DELETED.[CodInhumacion],DELETED.[CodIsam],DELETED.[NumeroPartidaDefuncion
],DELETED.[CodPropiedad],DELETED.[CodEmpleado],DELETED.[CausasFallecimie

```

```

nto],DELETED.[LugarFallecimiento],DELETED.[DireccionFallecimiento],DELETE
D.[FechaInhumacion], @TipoTransaccion, @now
        FROM DELETED
        END
END TRY

BEGIN CATCH
        ROLLBACK TRANSACTION
        RAISERROR ('HUBO UN ERROR EN LA INSERCIÓN',20,1)
END CATCH

--DISPARADOR [MODRIN].[TRGEXHUMACIONES]QUE SE UTILIZA PARA REALIZAR UN
LOG TRANSACCIONAL DE LAS PARTIDAS DE EXHUMACION.

USE [DBCEMENTERIO]
GO
/***** Object: Trigger [MODRIN].[TRGEXHUMACIONES]      Script Date:
04/06/2011 13:01:57 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [MODRIN].[TRGEXHUMACIONES] ON [MODRIN].[TBLEXHUMACIONES]
FOR INSERT,UPDATE,DELETE AS

DECLARE @now datetime,@TipoTransaccion varchar(10)

BEGIN TRY

SET @now = getdate()

        if exists (select * from inserted)
                if exists (select * from deleted)
                        select @TipoTransaccion = 'U'
                else
                        select @TipoTransaccion = 'I'
        else
                select @TipoTransaccion = 'D'

        IF (@TipoTransaccion='U' or @TipoTransaccion='I')
                BEGIN
                        INSERT INTO [MODRIN].[TBLEXHUMACIONES_AUDITORIA]
(CodExhumacion,CodInhumacion,FechaInhumacion,Observaciones,CausaInhumacio
n,CodEmpleado,Accion,FechaActualizacion)
                        SELECT
INSERTED.[CodExhumacion],INSERTED.[CodInhumacion],INSERTED.[FechaInhumaci
on],INSERTED.[Observaciones],INSERTED.[CausaInhumacion],INSERTED.[CodEmpl
eado], @TipoTransaccion, @now
                        FROM INSERTED
                END
        ELSE
                BEGIN

```

```

                INSERT INTO [MODRIN].[TBLEXHUMACIONES_AUDITORIA]
(CodExhumacion,CodInhumacion,FechaInhumacion,Observaciones,CausaInhumacio
n,CodEmpleado,Accion,FechaActualizacion)
                SELECT
DELETED.[CodExhumacion],DELETED.[CodInhumacion],DELETED.[FechaInhumacion]
,DELETED.[Observaciones],DELETED.[CausaInhumacion],DELETED.[CodEmpleado],
@TipoTransaccion, @now
                FROM DELETED
                END
END TRY

BEGIN CATCH
        ROLLBACK TRANSACTION
        RAISERROR ('HUBO UN ERROR EN LA INSERCIÓN',20,1)
END CATCH

--DISPARADOR [MODRIN].[TRGMSTPROPIEDAD] QUE SE UTILIZA PARA REALIZAR UN
LOG TRANSACCIONAL DE LAS PROPIEDADES.

USE [DBCEMENTERIO]
GO
/***** Object: Trigger [MODRIN].[TRGMSTPROPIEDAD]      Script Date:
04/24/2011 17:06:05 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [MODRIN].[TRGMSTPROPIEDAD] ON [MODRIN].[MSTPROPIEDAD] FOR
INSERT,UPDATE,DELETE AS

DECLARE @now datetime,@TipoTransaccion varchar(10)

BEGIN TRY

SET @now = getdate()

        if exists (select * from inserted)
                if exists (select * from deleted)
                        select @TipoTransaccion = 'U'
                else
                        select @TipoTransaccion = 'I'
        else
                select @TipoTransaccion = 'D'

        IF (@TipoTransaccion='U' or @TipoTransaccion='I')
                BEGIN
                        INSERT INTO [MODRIN].[MSTPROPIEDAD_AUDITORIA]
(CodPropiead,TipoPropiedad,CodIsam,CodTitulo,CodActa,CodCliente,CodEmplea
do,NombresComprador,ApellidosCompraador,OficioProfesionComprador,Telefono
Comprador,DireccionComprador,NumeroDuiComprador,LugarExtesnsionDuiComprad
or,FechaExtensionDuiComprador,Zona,Categoria,Cuadro,Calle,NoCalle,LetraCa
lle,PuestoSepultura,Fila,AnchoPropiead,LargoPropieadad,NumeroPuestoNorte,
NumeroCalleNorte,NumeroPuestoSur,NumeroCalleSur,NumeroPuestoOriente,Numer
oCalleOriente,NumeroPuestoPoniente,NumeroCallePoniente,NumeroNichosConstr
uir,FechaCompra,Nula,FechaNulidad,UsuarioAnulador,UsuarioAutoriza,Justifi
cacionNulidad,CodFirmas,Accion,FechaActualizacion)

```



```

        SELECT
INSERTED.CodPropiead, INSERTED.TipoPropiedad, INSERTED.CodIsam, INSERTED.Cod
Titulo, INSERTED.CodActa, INSERTED.CodCliente, INSERTED.CodEmpleado, INSERTED
.NombresComprador, INSERTED.ApellidosComprador, INSERTED.OficioProfesionCom
prador, INSERTED.TelefonoComprador, INSERTED.DireccionComprador, INSERTED.Nu
meroDuiComprador, INSERTED.LugarExtesnsionDuiComprador, INSERTED.FechaExten
sionDuiComprador, INSERTED.Zona, INSERTED.Categoria, INSERTED.Cuadro, INSETE
D.Calle, INSERTED.NoCalle, INSERTED.LetraCalle, INSERTED.PuestoSepultura, INS
ERTED.Fila, INSERTED.AnchoPropiead, INSERTED.LargoPropiedad, INSERTED.Numer
oPuestoNorte, INSERTED.NumeroCalleNorte, INSERTED.NumeroPuestoSur, INSERTED.
NumeroCalleSur, INSERTED.NumeroPuestoOriente, INSERTED.NumeroCalleOriente, I
NSERTED.NumeroPuestoPoniente, INSERTED.NumeroCallePoniente, INSERTED.Numero
NichosConstruir, INSERTED.FechaCompra, INSERTED.Nula, INSERTED.FechaNulidad,
INSERTED.UsuarioAnulador, INSERTED.UsuarioAutoriza, INSERTED.JustificacionN
ulidad, INSERTED.CodFirmas, @TipoTransaccion, @now
        FROM INSERTED
    END
ELSE
BEGIN
    INSERT INTO [MODRIN].[MSTPROPIEDAD_AUDITORIA]
(CodPropiead, TipoPropiedad, CodIsam, CodTitulo, CodActa, CodEmplea
do, NombresComprador, ApellidosCompraador, OficioProfesionComprador, Telefono
Comprador, DireccionComprador, NumeroDuiComprador, LugarExtesnsionDuiComprad
or, FechaExtensionDuiComprador, Zona, Categoria, Cuadro, Calle, NoCalle, LetraCa
lle, PuestoSepultura, Fila, AnchoPropiead, LargoPropiedad, NumeroPuestoNorte,
NumeroCalleNorte, NumeroPuestoSur, NumeroCalleSur, NumeroPuestoOriente, Numer
oCalleOriente, NumeroPuestoPoniente, NumeroCallePoniente, NumeroNichosConstr
uir, FechaCompra, Nula, FechaNulidad, UsuarioAnulador, UsuarioAutoriza, Justifi
cacionNulidad, CodFirmas, Accion, FechaActualizacion)
        SELECT
DELETED.CodPropiead, DELETED.TipoPropiedad, DELETED.CodIsam, DELETED.CodTitu
lo, DELETED.CodActa, DELETED.CodCliente, DELETED.CodEmpleado, DELETED.Nombres
Comprador, DELETED.ApellidosComprador, DELETED.OficioProfesionComprador, DEL
ETED.TelefonoComprador, DELETED.DireccionComprador, DELETED.NumeroDuiCompra
dor, DELETED.LugarExtesnsionDuiComprador, DELETED.FechaExtensionDuiComprado
r, DELETED.Zona, DELETED.Categoria, DELETED.Cuadro, DELETED.Calle, DELETED.NoC
alle, DELETED.LetraCalle, DELETED.PuestoSepultura, DELETED.Fila, DELETED.Anch
oPropiead, DELETED.LargoPropiedad, DELETED.NumeroPuestoNorte, DELETED.Numer
oCalleNorte, DELETED.NumeroPuestoSur, DELETED.NumeroCalleSur, DELETED.Numero
PuestoOriente, DELETED.NumeroCalleOriente, DELETED.NumeroPuestoPoniente, DEL
ETED.NumeroCallePoniente, DELETED.NumeroNichosConstruir, DELETED.FechaCompr
a, DELETED.Nula, DELETED.FechaNulidad, DELETED.UsuarioAnulador, DELETED.Usuar
ioAutoriza, DELETED.JustificacionNulidad, DELETED.CodFirmas, @TipoTransaccio
n, @now
        FROM DELETED
    END
END TRY

BEGIN CATCH
    ROLLBACK TRANSACTION
    RAISERROR ('HUBO UN ERROR EN LA INSERCIÓN', 20, 1)
END CATCH

```

PROCEDIMIENTOS ALMACENADOS

Acá se muestran todos los procedimientos almacenados que se encuentran en la base de datos, estos procedimientos son utilizados por las clases expuestas en los literales anteriores para poder establecer una comunicación entre la base de datos y la aplicación.

PROCEDIMIENTOS ALMACENADOS DEL MODULO DE USUARIOS (MODUSU)

A continuación se detallan uno a uno los procedimientos utilizados por las clases de este módulo.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Muestra los roles del usuario en base a su estado

CREATE PROCEDURE [MODUSU].[SP_VERROLES]

    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT          CodRol,
                   Nombre,
                   Descripcion
    FROM            MODUSU.CATROLES
    WHERE           Estado = @Estado
    ORDER BY       CodRol ASC
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Verifica si un usuario esta siendo ingresado dos veces

CREATE PROCEDURE [MODUSU].[SP_VERIFICARUSUARIO]

    @Usuario varchar(20)

AS
BEGIN

    SELECT          CodUsuario
    FROM            MODUSU.MSTUSUARIOS
    WHERE           Usuario = @Usuario
END
```

```

GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Comprueba si el usuario puede ingresar al sistema o no

CREATE PROCEDURE [MODUSU].[SP_USUARIOCORRECTO]

    @Usuario varchar(20),
    @Password varchar(128)

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      CodUsuario,
                Usuario,
                Password,
                Estado,
                Rol
    FROM        MODUSU.MSTUSUARIOS
    WHERE       Usuario = @Usuario AND
                Password = @Password AND
                Estado = 1

    ORDER BY   CodUsuario ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Modifica rol y estado del usuario

CREATE PROCEDURE [MODUSU].[SP_MODIFICARUSUARIO]

    @Codigo int,
    @Rol int,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE     MODUSU.MSTUSUARIOS
    SET        Rol = @Rol,
                Estado = @Estado
    WHERE      CodUsuario = @Codigo

END
GO

```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Modifica el password del usuario

CREATE PROCEDURE [MODUSU].[SP_MODIFICARPASSWORD]

    @Codigo int,
    @Password varchar(128)

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODUSU.MSTUSUARIOS
    SET Password = @Password
    WHERE CodUsuario = @Codigo
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra el usuario del empleado seleccionado

CREATE PROCEDURE [MODUSU].[SP_VERUSUARIO]

    @CodigoUsuario int

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      u.CodUsuario,
               u.Usuario,
               r.CodRol,
               u.Estado
    FROM        MODUSU.MSTUSUARIOS u,
               MODUSU.CATROLES r
    WHERE       u.Rol = r.CodRol AND
               u.CodUsuario = @CodigoUsuario

    ORDER BY   CodUsuario ASC
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

-- Author:      MeMuX
-- Description: Verifica si el empleado tiene un usuario activo

CREATE PROCEDURE [MODUSU].[SP_VERIFICARUSUARIOEMPLEADO]

    @CodigoEmpleado int

AS
BEGIN

    DECLARE    @CodigoUsuario int

    SET        @CodigoUsuario = (SELECT CodUsuario FROM
MODPER.MSTEMPLEADOS WHERE CodEmpleado = @CodigoEmpleado)

    SELECT     CodUsuario
FROM          MODUSU.MSTUSUARIOS
WHERE        CodUsuario = @CodigoUsuario AND
            Estado = 1

    ORDER BY  CodUsuario ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Agrega un nuevo usuario, tanto a la tabla de usuario,
asi como tmb se relaciona con el empleado

CREATE PROCEDURE [MODUSU].[SP_GUARDARUSUARIO]

    @Empleado int,
    @Usuario varchar(20),
    @Password varchar(128),
    @Rol int

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO  MODUSU.MSTUSUARIOS
                (Usuario,
                Password,
                Estado,
                Rol)
VALUES         (@Usuario,
                @Password,
                1,
                @Rol)

    UPDATE  MODPER.MSTEMPLEADOS
    SET     CodUsuario = (SELECT TOP 1 CodUsuario FROM MODUSU.MSTUSUARIOS
ORDER BY CodUsuario DESC)

```

```

WHERE CodEmpleado = @Empleado
END
GO

```

PROCEDIMIENTOS ALMACENADOS DEL MODULO DE REGISTRO DE INFORMACIÓN (MODRIN)

A través de estos procedimientos almacenados la capa de datos puede interactuar con la base de datos mandándole todos los datos que se deseen almacenar.

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_VERIFICARPROPIEDAD]
Script Date: 04/24/2011 17:13:18 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VERIFICARPROPIEDAD]
@TipoPropiedad varchar(50),
@CodIsam int,
@CodTitulo int

AS
BEGIN
    SELECT MODRIN.MSTPROPIEDADAD.CodTitulo FROM
    [DBCEMENTERIO].[MODRIN].[MSTPROPIEDADAD] WHERE
    TipoPropiedad=@TipoPropiedad and CodIsam=@CodIsam and
    CodTitulo=@CodTitulo and Nula='False'
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_VERIFICARPATIDADEFUNCION]
Script Date: 04/24/2011 17:13:16 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VERIFICARPATIDADEFUNCION]

@NumeroPartidaDefuncion varchar(6),
@CodIsam int

AS
BEGIN
    SELECT MODRIN.TBLPARTIDADEFUNCION.NumeroPartidaDefuncion FROM
    MODRIN.TBLPARTIDADEFUNCION WHERE
    NumeroPartidaDefuncion=@NumeroPartidaDefuncion AND CodIsam=@CodIsam
END
USE [DBCEMENTERIO]
GO

```

```

/***** Object: StoredProcedure [MODRIN].[SP_VERIFICARCLIENTE]      Script
Date: 04/24/2011 17:13:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VERIFICARCLIENTE]
@NumeroDui varchar(50)
AS
BEGIN
    SELECT NumeroDui, Nombres, Apellidos FROM MODRIN.MSTCLIENTES WHERE
NumeroDui=@NumeroDui
END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_VERIFICARBENEFICIARIO]
Script Date: 04/24/2011 17:13:09 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VERIFICARBENEFICIARIO]

    @CodPropiedad int,
    @NumeroDui varchar(50)

AS
BEGIN

    SELECT      CodBeneficiario
FROM MODRIN.MSTBENEFICIARIOS
WHERE CodPropiedad = @CodPropiedad AND
        NumeroDui = @NumeroDui

END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_VALORTIPONICHO]      Script
Date: 04/24/2011 17:13:06 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VALORTIPONICHO]
@TipoNicho varchar(20)
AS
BEGIN
    SELECT      SUM(Valor) AS TotalServicio
FROM DBCEMENTERIO.MODCYC.MSTSERVICIO
WHERE Nombre=@TipoNicho
END
USE [DBCEMENTERIO]

```

```

GO
/***** Object: StoredProcedure [MODRIN].[SP_VALORFRMNICHO]      Script
Date: 04/24/2011 17:13:02 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VALORFRMNICHO]

AS
BEGIN
    SELECT          SUM(Valor) AS ValorFormulario
    FROM    DBCEMENTERIO.MODCYC.MSTSERVICIO
    WHERE Nombre='FORMULARIO NICHO'
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_VALORFF]      Script Date:
04/24/2011 17:13:00 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_VALORFF]

AS
BEGIN
    SELECT          Valor AS FF
    FROM    DBCEMENTERIO.MODCYC.MSTIMPUESTO
    WHERE Nombre='FONDO DE FIESTAS'
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_OBTENERNOMBREFALLECIDO]
Script Date: 04/24/2011 17:12:57 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_OBTENERNOMBREFALLECIDO]
    @NumeroPartidaDefuncion int

AS
BEGIN

SELECT          NombreFallecido, OrigenFallecido, DireccionFallecido,
NombrePadre, NombreMadre, MedicoAvalaDefuncion,HoraFechaFallecimiento,
Genero
FROM            MODRIN.TBLPARTIDADEFUNCION
WHERE          NumeroPartidaDefuncion=@NumeroPartidaDefuncion

END

```



```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_OBTENERCODIGOPROPIEDAD]
Script Date: 04/24/2011 17:12:54 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Obtiene el codigo de la propiedad en base a ciertos
parametros

```

```

ALTER PROCEDURE [MODRIN].[SP_OBTENERCODIGOPROPIEDAD]

```

```

    @Cliente int,
    @ISAM int,
    @Titulo int

```

```

AS

```

```

BEGIN

```

```

    SET NOCOUNT ON;

```

```

    SELECT CodPropiead
    FROM MODRIN.MSTPROPIEDAD

```

```

    WHERE CodCliente = @Cliente AND CodIsam = @ISAM AND CodTitulo =

```

```

@Titulo

```

```

END

```

```

USE [DBCEMENTERIO]

```

```

GO

```

```

/***** Object: StoredProcedure [MODRIN].[SP_MODIFICARCLIENTES]

```

```

Script Date: 04/24/2011 17:12:49 *****/

```

```

SET ANSI_NULLS ON

```

```

GO

```

```

SET QUOTED_IDENTIFIER ON

```

```

GO

```

```

ALTER PROCEDURE [MODRIN].[SP_MODIFICARCLIENTES]

```

```

    @Nombres varchar(250),
    @Apellidos varchar(250),
    @EstadoCivil varchar(1),
    @Direccion varchar(350),
    @NumeroDui varchar(50),
    @LugarExtensionDui varchar(50),
    @DptoExtDui varchar(150),
    @FechaNacimiento datetime,
    @Telefono varchar(9),
    @CodCliente int

```

```

AS

```

```

BEGIN

```

```

    UPDATE [DBCEMENTERIO].[MODRIN].[MSTCLIENTES]

```

```

    SET [Nombres] = @Nombres
    , [Apellidos] = @Apellidos

```

```

    , [EstadoCivil] = @EstadoCivil
    , [Direccion] = @Direccion
    , [NumeroDui] = @NumeroDui
    , [LugarExtensionDui] = @LugarExtensionDui
    , [DptoExtDui]=@DptoExtDui
    , [FechaNacimiento] = @FechaNacimiento
    , [Telefono] = @Telefono

    WHERE [CodCliente]=@CodCliente

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_MODIFICARBENEFICIARIO]
Script Date: 04/24/2011 17:12:46 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Modifica el beneficiario seleccionado

ALTER PROCEDURE [MODRIN].[SP_MODIFICARBENEFICIARIO]

    @CodBeneficiario int,
    @CodPropiedad int,
    @Nombres varchar(250),
    @Apellidos varchar(250),
    @NumeroDui varchar(50)

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODRIN.MSTBENEFICIARIOS
    SET      CodPropiedad = @CodPropiedad,
            Nombres = @Nombres,
            Apellidos = @Apellidos,
            NumeroDui = @NumeroDui
    WHERE   CodBeneficiario = @CodBeneficiario
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARPROPIEDADFALLECIDOS]
Script Date: 04/24/2011 17:12:43 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra todas las propiedades con fallecidos

```

```

ALTER PROCEDURE [MODRIN].[SP_LISTARPROPIEDADFALLECIDOS]

AS
BEGIN

    SELECT p.CodPropiead, p.TipoPropiedad, p.NombresComprador + ' ' +
p.ApellidosComprador AS NombreCompleto, p.Zona, p.Categoria,
    p.Cuadro, p.Calle, p.PuestoSepultura, d.NumeroPartidaDefuncion,
d.NombreFallecido, d.HoraFechaFallecimiento
    FROM MODRIN.MSTPROPIEADAD p, MODRIN.TBLINHUMACION i,
MODRIN.TBLPARTIDADEFUNCION d
    WHERE p.CodPropiead = i.CodPropiead AND i.NumeroPartidaDefuncion
= d.NumeroPartidaDefuncion
    ORDER BY p.CodPropiead ASC
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure
[MODRIN].[SP_LISTARPROPIEDADBENEFICIARIOS]      Script Date: 04/24/2011
17:12:41 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra todas las propiedades con beneficiarios

ALTER PROCEDURE [MODRIN].[SP_LISTARPROPIEDADBENEFICIARIOS]

AS
BEGIN

    SELECT p.CodPropiead, p.TipoPropiedad, p.NombresComprador + ' ' +
p.ApellidosComprador AS NombreCompleto, p.Zona, p.Categoria,
    p.Cuadro, p.Calle, p.PuestoSepultura, b.Nombres + ' ' + b.Apellidos
AS Beneficiarios, b.NumeroDui
    FROM MODRIN.MSTPROPIEADAD p, MODRIN.MSTBENEFICIARIOS b
    WHERE p.CodPropiead =b.CodPropiedad
    ORDER BY p.CodPropiead ASC
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARPROPIEDAD]      Script
Date: 04/24/2011 17:12:37 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_LISTARPROPIEDAD]

AS

```

```

BEGIN

    SELECT CodPropiead, TipoPropiedad, REPLACE(NombresComprador, ' ',
    ''') + ' ' + REPLACE(ApellidosComprador, ' ', ''') AS NombreCompleto,
    Zona, Categoria, Cuadro, Calle, PuestoSepultura
    FROM MODRIN.MSTPROPIEDAD
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARISAM]      Script Date:
04/24/2011 17:12:34 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_LISTARISAM]

AS
BEGIN

SELECT
    MODRIN.MSTCLIENTES.Nombres+ ' ' +MODRIN.MSTCLIENTES.Apellidos
as Cliente,
    MODCYC.MSTISAM.NumeroIsam,
    MODCYC.MSTISAM.CodIsam,
    MODCYC.MSTISAM.Isdem,
    MODCYC.MSTISAM.Serie,
    MODCYC.MSTISAM.Fecha,
    MODCYC.MSTISAM.Total

FROM MODCYC.MSTISAM,MODRIN.MSTCLIENTES

WHERE MODCYC.MSTISAM.CodCliente=MODRIN.MSTCLIENTES.CodCliente and
MODCYC.MSTISAM.Estado='True'

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARINHUMACIONES]
Script Date: 04/24/2011 17:12:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Lista todas las inhumaciones

ALTER PROCEDURE [MODRIN].[SP_LISTARINHUMACIONES]

AS
BEGIN

```

```

        SELECT i.CodInhumacion, p.NumeroPartidaDefuncion,
p.NombreFallecido, p.NombrePadre, p.NombreMadre,
p.HoraFechaFallecimiento, i.FechaInhumacion
        FROM MODRIN.TBLPARTIDADEFUNCION p, MODRIN.TBLINHUMACION i
        WHERE i.NumeroPartidaDefuncion = p.NumeroPartidaDefuncion
        ORDER BY i.CodInhumacion ASC
END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARDETALLEISAM]
Script Date: 04/24/2011 17:12:28 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_LISTARDETALLEISAM]
@CodIsam int,
@Isdem int,
@Serie varchar(2),
@NumeroIsam int
AS
BEGIN

SELECT DISTINCT
                MODCYC.TBLDETALLEISAM.CodIsam,
MODCYC.MSTSERVICIO.Nombre AS NombreServicio, MODCYC.MSTIMPUESTO.Nombre AS
NombreImpuesto

FROM
                MODCYC.MSTIMPUESTO INNER JOIN
                MODCYC.TBLDETALLEISAM ON
MODCYC.MSTIMPUESTO.CodImpuesto = MODCYC.TBLDETALLEISAM.CodImpuesto INNER
JOIN
                MODCYC.MSTSERVICIO ON
MODCYC.TBLDETALLEISAM.CodServicio = MODCYC.MSTSERVICIO.CodServicio INNER
JOIN
                MODCYC.MSTISAM ON MODCYC.TBLDETALLEISAM.CodIsam =
MODCYC.MSTISAM.CodIsam

WHERE
                (MODCYC.TBLDETALLEISAM.CodIsam = @CodIsam) and
(MODCYC.MSTISAM.NumeroIsam=@NumeroIsam) and (MODCYC.MSTISAM.Isdem=@Isdem)
and (MODCYC.MSTISAM.Serie=@Serie)

END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARDEFUNCIONES]
Script Date: 04/24/2011 17:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_LISTARDEFUNCIONES]

```

```

AS
BEGIN

    SELECT NumeroPartidaDefuncion, NombreSolicitante, NombreFallecido,
    HoraFechaFallecimiento, NombrePadre, NombreMadre
    FROM MODRIN.TBLPARTIDADEFUNCION
    ORDER BY NumeroPartidaDefuncion ASC
END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARCLIENTES]      Script
Date: 04/24/2011 17:12:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_LISTARCLIENTES]

```

```

AS
BEGIN

    SELECT      Nombres+ ' ' +Apellidos AS NombreCompleto,CodCliente,
    Nombres, Apellidos,
                EstadoCivil, Direccion, NumeroDui,
    LugarExtensionDui, DptoExtDui ,FechaNacimiento, Telefono
    FROM        MODRIN.MSTCLIENTES
END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_LISTARBENEFICIARIOS]
Script Date: 04/24/2011 17:12:19 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra los beneficiarios por propiedad

ALTER PROCEDURE [MODRIN].[SP_LISTARBENEFICIARIOS]

```

```
@Propiedad int
```

```

AS
BEGIN

SET NOCOUNT ON;

    SELECT      CodBeneficiario, Nombres, Apellidos, NumeroDui, Estado
    FROM MODRIN.MSTBENEFICIARIOS
    WHERE CodPropiedad = @Propiedad
END

```

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_INSERTARCLIENTES]      Script
Date: 04/24/2011 17:12:16 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_INSERTARCLIENTES]

    @Nombres varchar(250)
    ,@Apellidos varchar(250)
    ,@EstadoCivil varchar(1)
    ,@Direccion varchar(350)
    ,@NumeroDui varchar(50)
    ,@LugarExtensionDui varchar(50)
    ,@FechaNacimiento datetime
    ,@Telefono varchar(9)

AS
BEGIN

INSERT INTO [DBCEMENTERIO].[MODRIN].[MSTCLIENTES]
    ([Nombres]
    , [Apellidos]
    , [EstadoCivil]
    , [Direccion]
    , [NumeroDui]
    , [LugarExtensionDui]
    , [FechaNacimiento]
    , [Telefono])

VALUES

    (@Nombres
    , @Apellidos
    , @EstadoCivil
    , @Direccion
    , @NumeroDui
    , @LugarExtensionDui
    , @FechaNacimiento
    , @Telefono)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_INSERTARBENEFICIARIOS]
Script Date: 04/24/2011 17:12:13 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_INSERTARBENEFICIARIOS]

    @CodPropiedad int,
    @Nombres varchar(250),
    @Apellidos varchar(250),
    @NumeroDui varchar(50)

```

```

AS
BEGIN

    INSERT INTO [DBCEMENTERIO].[MODRIN].[MSTBENEFICIARIOS]
        ([CodPropiedad]
        , [Nombres]
        , [Apellidos]
        , [NumeroDui])
    VALUES
        (@CodPropiedad
        , @Nombres
        , @Apellidos
        , @NumeroDui)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_GUARDARTESTIGO]      Script
Date: 04/24/2011 17:12:10 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_GUARDARTESTIGO]
    @NumeroPartidaDefuncion varchar(6),
    @Nombres varchar(250),
    @Apellidos varchar(250),
    @NumeroDui varchar(10),
    @ParentescoFallecido varchar(50)
AS
BEGIN
    INSERT INTO [DBCEMENTERIO].[MODRIN].[MSTTESTIGOS]
        ([NumeroPartidaDefuncion]
        , [Nombres]
        , [Apellidos]
        , [NumeroDui]
        , [ParentescoFallecido])
    VALUES
        (@NumeroPartidaDefuncion,
        @Nombres,
        @Apellidos,
        @NumeroDui,
        @ParentescoFallecido)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_GUARDARPROPIEDAD]    Script
Date: 04/24/2011 17:12:08 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```



```

ALTER PROCEDURE [MODRIN].[SP_GUARDARPROPIEDAD]

@TipoPropiedad varchar(50),
@CodIsam int,
@CodTitulo int,
@CodActa int,
@CodCliente int,
@CodEmpleado int,
@NombresComprador varchar(250),
@ApellidosComprador varchar(250),
@OficioProfesionComprador varchar(250),
@TelefonoComprador varchar(9),
@DireccionComprador varchar(350),
@NumeroDuiComprador varchar(10),
@LugarExtesnsionDuiComprador varchar(50),
@FechaExtensionDuiComprador datetime,
@Zona varchar(50),
@Categoria int,
@Cuadro varchar(50),
@Calle varchar(50),
@NoCalle varchar(10),
@LetraCalle varchar(10),
@PuestoSepultura int,
@Fila int,
@AnchoPropiead float,
@LargoPropiedad float,
@NumeroPuestoNorte int,
@NumeroCalleNorte int,
@NumeroPuestoSur int,
@NumeroCalleSur int,
@NumeroPuestoOriente int,
@NumeroCalleOriente int,
@NumeroPuestoPoniente int,
@NumeroCallePoniente int,
@NumeroNichosConstruir int,
@FechaCompra datetime,
@CodFirmas int,
@DptoExtDui varchar(150)
AS
BEGIN

IF NOT EXISTS (SELECT MODRIN.MSTPROPIEADAD.CodTitulo FROM
[DBCEMENTERIO].[MODRIN].[MSTPROPIEADAD] WHERE
TipoPropiedad=@TipoPropiedad and CodIsam=@CodIsam and
CodTitulo=@CodTitulo and Nula='False')
BEGIN
INSERT INTO [DBCEMENTERIO].[MODRIN].[MSTPROPIEADAD]
([TipoPropiedad]
,[CodIsam]
,[CodTitulo]
,[CodActa]
,[CodCliente]
,[CodEmpleado]
,[NombresComprador]
,[ApellidosComprador]
,[OficioProfesionComprador]

```

```

, [TelefonoComprador]
, [DireccionComprador]
, [NumeroDuiComprador]
, [LugarExtesnsionDuiComprador]
, [DptoExtDui]
, [FechaExtensionDuiComprador]
, [Zona]
, [Categoria]
, [Cuadro]
, [Calle]
, [NoCalle]
, [LetraCalle]
, [PuestoSepultura]
, [Fila]
, [AnchoPropiead]
, [LargoPropiedad]
, [NumeroPuestoNorte]
, [NumeroCalleNorte]
, [NumeroPuestoSur]
, [NumeroCalleSur]
, [NumeroPuestoOriente]
, [NumeroCalleOriente]
, [NumeroPuestoPoniente]
, [NumeroCallePoniente]
, [NumeroNichosConstruir]
, [FechaCompra]
, [Nula]
, [CodFirmas])

```

VALUES

```

(@TipoPropiedad,
@CodIsam,
@CodTitulo,
@CodActa,
@CodCliente,
@CodEmpleado,
@NombresComprador,
@ApellidosComprador,
@OficioProfesionComprador,
@TelefonoComprador,
@DireccionComprador,
@NumeroDuiComprador,
@LugarExtesnsionDuiComprador,
@DptoExtDui,
@FechaExtensionDuiComprador,
@Zona,
@Categoria,
@Cuadro,
@Calle,
@NoCalle,
@LetraCalle,
@PuestoSepultura,
@Fila,
@AnchoPropiead,
@LargoPropiedad,
@NumeroPuestoNorte,
@NumeroCalleNorte,

```

```

        @NumeroPuestoSur,
        @NumeroCalleSur,
        @NumeroPuestoOriente,
        @NumeroCalleOriente,
        @NumeroPuestoPoniente,
        @NumeroCallePoniente,
        @NumeroNichosConstruir,
        @FechaCompra,
        'False',
        @CodFirmas)
    END
END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_GUARDARPARTIDADEFUNCION]
Script Date: 04/24/2011 17:12:05 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_GUARDARPARTIDADEFUNCION]

@NumeroPartidaDefuncion varchar(6),
@CodIsam int,
@NombreSolicitante varchar(250),
@NombreFallecido varchar(250),
@Edad int,
@EstadoCivil varchar(1),
@OrigenFallecido varchar(50),
@DireccionFallecido varchar(350),
@OficioProfesionFallecidos varchar(250),
@NombrePadre varchar(250),
@NombreMadre varchar(250),
@MedicoAvalaDefuncion varchar(250),
@HoraFechaFallecimiento datetime,
@CodEmpleado int,
@TipoFallecimiento varchar(50),
@CausaMuerte varchar(150),
@Genero varchar(12)
AS
BEGIN
    INSERT INTO [DBCEMENTERIO].[MODRIN].[TBLPARTIDADEFUNCION]
        ([NumeroPartidaDefuncion]
        , [CodIsam]
        , [NombreSolicitante]
        , [NombreFallecido]
        , [Edad]
        , [EstadoCivil]
        , [OrigenFallecido]
        , [DireccionFallecido]
        , [OficioProfesionFallecidos]
        , [NombrePadre]
        , [NombreMadre]
        , [MedicoAvalaDefuncion]
        , [HoraFechaFallecimiento]

```

```

        , [CodEmpleado]
        , [TipoFallecimiento]
        , [CausaMuerte]
        , [Genero])
VALUES
    (@NumeroPartidaDefuncion,
    @CodIsam,
    @NombreSolicitante,
    @NombreFallecido,
    @Edad,
    @EstadoCivil,
    @OrigenFallecido,
    @DireccionFallecido,
    @OficioProfesionFallecidos,
    @NombrePadre,
    @NombreMadre,
    @MedicoAvalaDefuncion,
    @HoraFechaFallecimiento,
    @CodEmpleado,
    @TipoFallecimiento,
    @CausaMuerte,
    @Genero)

END

USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODRIN].[SP_GUARDARINHUMACION]
Script Date: 04/24/2011 17:12:03 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Guarda una nueva inhumación

ALTER PROCEDURE [MODRIN].[SP_GUARDARINHUMACION]

    @CodIsam int, @NumeroPartidaDefuncion varchar(6), @CodPropiedad
int,
    @CodEmpleado int, @CausasFallecimiento varchar(500),
@LugarFallecimiento varchar(50),
    @DireccionFallecimiento varchar(250), @FechaInhumacion Datetime
AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO MODRIN.TBLINHUMACION(CodIsam, NumeroPartidaDefuncion,
CodPropiedad, CodEmpleado, CausasFallecimiento,
LugarFallecimiento, DireccionFallecimiento, FechaInhumacion)
VALUES (@CodIsam, @NumeroPartidaDefuncion, @CodPropiedad,
@CodEmpleado, @CausasFallecimiento,
@LugarFallecimiento, @DireccionFallecimiento, @FechaInhumacion)
END

USE [DBCEMENTERIO]

```

```

GO
/***** Object: StoredProcedure [MODRIN].[SP_GUARDARFIRMAS]      Script
Date: 04/24/2011 17:12:00 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_GUARDARFIRMAS]

    @FirmaAlcalde varchar(250),
    @FirmaSecretario varchar(250),
    @FirmaAdministrador varchar(250),
    @FechaIngreso datetime,
    @CodUsuario int,
    @Activa bit

AS
BEGIN
    INSERT INTO [DBCEMENTERIO].[MODRIN].[CATFIRMAS]
        ([FirmaAlcalde]
        , [FirmaSecretario]
        , [FirmaAdministrador]
        , [FechaIngreso]
        , [CodUsuario]
        , [Activa])
    VALUES
        (@FirmaAlcalde
        , @FirmaSecretario
        , @FirmaAdministrador
        , @FechaIngreso
        , @CodUsuario
        , @Activa)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_GUARDAREXHUMACION]
Script Date: 04/24/2011 17:11:52 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Guarda una nueva exhumación

ALTER PROCEDURE [MODRIN].[SP_GUARDAREXHUMACION]
    @CodInhumacion int, @FechaInhumacion datetime, @Observaciones
varchar(350),
    @CausaInhumacion varchar(50), @CodEmpleado int

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO MODRIN.TBLEXHUMACIONES( CodInhumacion, FechaInhumacion,
Observaciones, CausaInhumacion, CodEmpleado)

```

```

VALUES ( @CodInhumacion, @FechaInhumacion, @Observaciones,
@CausaInhumacion, @CodEmpleado)
END

USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODRIN].[SP_GUARDARCLIENTE]      Script
Date: 04/24/2011 17:11:49 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_GUARDARCLIENTE]
@Nombres varchar(250),
@Apellidos varchar(250),
@EstadoCivil varchar(1),
@Direccion varchar(350),
@NumeroDui varchar(50),
@LugarExtensionDui varchar(50),
@DptoExtDui varchar(150),
@FechaNacimiento datetime,
@Telefono varchar(9)

AS
BEGIN

INSERT INTO [DBCEMENTERIO].[MODRIN].[MSTCLIENTES]
([Nombres]
,[Apellidos]
,[EstadoCivil]
,[Direccion]
,[NumeroDui]
,[LugarExtensionDui]
,[DptoExtDui]
,[FechaNacimiento]
,[Telefono])
VALUES
(@Nombres,
@Apellidos,
@EstadoCivil,
@Direccion,
@NumeroDui,
@LugarExtensionDui,
@DptoExtDui,
@FechaNacimiento,
@Telefono)

END

USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODRIN].[SP_GUARDARBENEFICIARIO]
Script Date: 04/24/2011 17:11:46 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

-- Author:          <Author,,Name>
-- Description:     <Description,,>

ALTER PROCEDURE [MODRIN].[SP_GUARDARBENEFICIARIO]

    @CodPropiedad int,
    @Nombres varchar(250),
    @Apellidos varchar(250),
    @NumeroDui varchar(50)

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO MODRIN.MSTBENEFICIARIOS(CodPropiedad, Nombres, Apellidos,
NumeroDui, Estado)
    VALUES (@CodPropiedad, @Nombres, @Apellidos, @NumeroDui, 1)
END

USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODRIN].[SP_FIRMASACTIVAS]      Script
Date: 04/24/2011 17:11:42 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_FIRMASACTIVAS]

AS
BEGIN
    SELECT          [CodFirmas]
                  , [FirmaAlcalde]
                  , [FirmaSecretario]
                  , [FirmaAdministrador]
                  , [FechaIngreso]
                  , [CodUsuario]
                  , [Activa]
    FROM [DBCEMENTERIO].[MODRIN].[CATFIRMAS]
    WHERE Activa='True'
END

USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODRIN].[SP_ESTADISTICASDEFUNCIONES]
Script Date: 04/24/2011 17:11:40 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ESTADISTICASDEFUNCIONES]
    @FechaInicio varchar(25),
    @FechaFin varchar(25)

AS
BEGIN

```

```

SELECT      COUNT(NumeroPartidaDefuncion) AS DEFUNCIONES
FROM        MODRIN.TBLPARTIDADEFUNCION
WHERE       (HoraFechaFallecimiento BETWEEN @FechaInicio AND @FechaFin)

END

USE [DBCENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ESTADISTICAPROPIEDAD]
Script Date: 04/24/2011 17:11:37 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ESTADISTICAPROPIEDAD]
    @FechaInicio varchar(25),
    @FechaFin varchar(25)
AS
BEGIN

SELECT      COUNT(TipoPropiedad) AS PROPIEDADES
FROM        MODRIN.MSTPROPIEDAD
WHERE       (FechaCompra BETWEEN @FechaInicio AND @FechaFin) AND (Nula =
'False')

END

USE [DBCENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ESTADISTICAINHUMACIONES]
Script Date: 04/24/2011 17:11:33 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ESTADISTICAINHUMACIONES]
    @FechaInicio varchar(25),
    @FechaFin varchar(25)
AS
BEGIN

SELECT      COUNT(CodInhumacion) AS INHUMACIONES
FROM        MODRIN.TBLINHUMACION
WHERE       (FechaInhumacion BETWEEN @FechaInicio AND @FechaFin)

END

USE [DBCENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ESTADISTICAEXHUMACIONES]
Script Date: 04/24/2011 17:11:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```



```

ALTER PROCEDURE [MODRIN].[SP_ESTADISTICAEXHUMACIONES]
    @FechaInicio varchar(25),
    @FechaFin varchar(25)
AS
BEGIN

SELECT      COUNT(CodExhumacion) AS EXHUMACIONES
FROM        MODRIN.TBLEXHUMACIONES
WHERE       (FechaInhumacion BETWEEN @FechaInicio AND @FechaFin)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ELIMINARBENEFICIARIOS]
Script Date: 04/24/2011 17:11:28 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ELIMINARBENEFICIARIOS]
    @CodBeneficiario int,
    @CodPropiedad int
AS
BEGIN

    UPDATE [DBCEMENTERIO].[MODRIN].[MSTBENEFICIARIOS]

        SET     [Estado] = 'False'

        WHERE [CodBeneficiario] = @CodBeneficiario and
[CodPropiedad]=@CodPropiedad

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_CONTROLEXHUMACIONES]
Script Date: 04/24/2011 17:11:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_CONTROLEXHUMACIONES]
    @FechaInicio varchar(25),
    @FechaFin varchar(25)
AS
BEGIN

SELECT      MODRIN.TBLEXHUMACIONES.FechaInhumacion,
MODRIN.TBLPARTIDADEFUNCION.NombreFallecido, MODRIN.MSTPROPIEDAD.Cuadro,

```

```

MODRIN.MSTPROPIEDAD.Fila, MODRIN.MSTPROPIEDAD.Calle,
MODRIN.MSTPROPIEDAD.PuestoSepultura,
MODRIN.TBLEXHUMACIONES.Observaciones

FROM      MODRIN.TBLEXHUMACIONES INNER JOIN
           MODRIN.TBLINHUMACION ON MODRIN.TBLEXHUMACIONES.CodInhumacion =
MODRIN.TBLINHUMACION.CodInhumacion INNER JOIN
           MODRIN.TBLPARTIDADEFUNCION ON
MODRIN.TBLINHUMACION.NumeroPartidaDefuncion =
MODRIN.TBLPARTIDADEFUNCION.NumeroPartidaDefuncion INNER JOIN
           MODRIN.MSTPROPIEDAD ON MODRIN.TBLINHUMACION.CodPropiedad =
MODRIN.MSTPROPIEDAD.CodPropiead

WHERE     (MODRIN.TBLEXHUMACIONES.FechaInhumacion BETWEEN @FechaInicio
AND @FechaFin)

END

USE [DBCENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_CODFIRMASACTIVAS]      Script
Date: 04/24/2011 17:11:22 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_CODFIRMASACTIVAS]
AS
BEGIN
    SELECT CodFirmas FROM MODRIN.CATFIRMAS WHERE Activa='True'
END

USE [DBCENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_BUSCARPROPIEDADCODIGO]
Script Date: 04/24/2011 17:11:20 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_BUSCARPROPIEDADCODIGO]
@TipoPropiedad varchar(50),
@CodTitulo int
AS
BEGIN

SELECT      CodPropiead,TipoPropiedad,CodIsam,CodActa, CodCliente,
CodEmpleado, NombresComprador, ApellidosComprador,
OficioProfesionComprador, TelefonoComprador, DireccionComprador,
NumeroDuiComprador, LugarExtesnsionDuiComprador,
FechaExtensionDuiComprador, Zona, Categoria, Cuadro, Calle, NoCalle,
LetraCalle, PuestoSepultura, Fila, AnchoPropiead,
NumeroPuestoNorte, LargoPropiedad, NumeroCalleNorte,
NumeroPuestoSur, NumeroCalleSur, NumeroPuestoOriente, NumeroCalleOriente,
NumeroPuestoPoniente,

```

```

        NumeroCallePoniente, NumeroNichosConstruir,
FechaCompra, CodPropiead, CodTitulo, DptoExtDui, CodFirmas
FROM      MODRIN.MSTPROPIEDAD
WHERE     (Nula = 'False') AND (TipoPropiedad = @TipoPropiedad) AND
(CodTitulo = @CodTitulo)

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ANULARTITULOPROPIEDAD]
Script Date: 04/24/2011 17:11:17 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ANULARTITULOPROPIEDAD]

@Nula bit,
@FechaNulidad datetime,
@UsuarioAnulador int,
@UsuarioAutoriza int,
@JustificacionNulidad varchar(250),
@CodPropiedad int,
@TipoPropiedad varchar(50)

AS
BEGIN

    UPDATE [DBCEMENTERIO].[MODRIN].[MSTPROPIEDAD]
    SET     [Nula] = @Nula
           , [FechaNulidad] = @FechaNulidad
           , [UsuarioAnulador] = @UsuarioAnulador
           , [UsuarioAutoriza] = @UsuarioAutoriza
           , [JustificacionNulidad] = @JustificacionNulidad

    WHERE  CodPropiead=@CodPropiedad and TipoPropiedad=@TipoPropiedad

END

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODRIN].[SP_ANULARFIRMAS]      Script
Date: 04/24/2011 17:11:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODRIN].[SP_ANULARFIRMAS]
@CodFirmas int
AS
BEGIN
    UPDATE [DBCEMENTERIO].[MODRIN].[CATFIRMAS]
    SET [Activa] = 'False'
    WHERE CodFirmas=@CodFirmas
END

```

PROCEDIMIENTOS ALMACENADOS DEL MODULO DE PERSONAL (MODPER)

A continuación se detallan uno a uno los procedimientos utilizados por las clases de este módulo.

```
SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_INSERTARDEPARTAMENTO]
    @Nombre varchar(250)

AS
BEGIN

    SETNOCOUNTON;

    INSERTINTO [MODPER].CATDEPARTAMENTO (Nombre)
    VALUES (@Nombre)
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_INSERTARCONTRATISTA]
    @Nombre varchar(250),
    @Apellidos varchar(250),
    @Direcc varchar(250),
    @Tel varchar(9),
    @Du varchar(10),
    @Ni varchar(17),
    @Estad bit

AS
BEGIN

    SETNOCOUNTON;

    INSERTINTO [MODPER].MSTCONTRATISTA (Nombres, Apellidos, Direccion,
    Telefono, Dui, Nit, Estado)
    VALUES (@Nombre, @Apellidos, @Direcc, @Tel, @Du, @Ni, @Estad)
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_VERDEPARTAMENTOS]

AS
BEGIN
```

```

        SETNOCOUNTON;

SELECT CodDepto, Nombre As [Nombre de los Departamentos]
FROM DBCEMENTERIO.MODPER.CATDEPARTAMENTO
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_VERCONTRATISTAS]

AS
BEGIN

        SETNOCOUNTON;

SELECT CodContratista, Nombres, Apellidos, Direccion, Telefono, Dui, Nit,
Estado
FROM DBCEMENTERIO.MODPER.MSTCONTRATISTA
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_MODIFICARDEPARTAMENTO]

        @cod int,
        @name varchar(250)

AS
BEGIN

        SETNOCOUNTON;

UPDATE [MODPER].CATDEPARTAMENTO
SET NOmbre=@name
WHERE CodDepto=@Cod
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_MODIFICARCONTRATISTA]

        @codido int,
        @name varchar(250),
        @ape varchar(250),
        @address varchar(250),
        @tel varchar(9),

```

```

        @du varchar(10),
        @ni varchar(17),
        @estad bit

AS
BEGIN

        UPDATE [MODPER].MSTCONTRATISTA
        SET Nombres=@name, Apellidos=@ape, Direccion=@address,
        Telefono=@tel, Dui=@du, Nit=@ni, Estado=@estad
        WHERE CodContratista=@codido

END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_MOSTRARDEPARTAMENTOS]

AS
BEGIN

        SETNOCOUNTON;

SELECT Nombre
FROM [MODPER].CATDEPARTAMENTO
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_VEREMPLEADOS]

        @Estado bit

AS
BEGIN

        SETNOCOUNTON;

SELECT CodEmpleado, Nombres, Apellidos
FROM MODPER.MSTEMPLEADOS
WHERE Estado = @Estado
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_MODIFICAREMPLEADO]

        @codigo int,

```

```

@NombreDepto varchar(250),
@name varchar(250),
@ape varchar(250),
@address varchar(350),
@tel varchar(9),
@du varchar(10),
@ni varchar(17),
@estad bit

AS
BEGIN

    SETNOCOUNTON;

    DECLARE @CodDep int
    SET @CodDep =(SELECT CodDepto FROM
DBCEMENTERIO.MODPER.CATDEPARTAMENTO WHERE Nombre=@NombreDepto)

UPDATE [MODPER].MSTEMPLEADOS
SET CodDepto=@CodDep, Nombres=@name, Apellidos=@ape, Direccion=@address,
Telefono=@tel, Dui=@du, Nit=@ni, Estado=@estad
WHERE CodEmpleado=@codigo
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_INSERTAREMPLEADO]

@NombreDepto varchar(250),
@Nom varchar(250),
@Ape varchar(250),
@Direcc varchar(350),
@Tel varchar(9),
@Du varchar(10),
@Ni varchar(17),
@Estad bit

AS
BEGIN

    SETNOCOUNTON;

    DECLARE @CodDep int
    SET @CodDep =(SELECT CodDepto FROM
DBCEMENTERIO.MODPER.CATDEPARTAMENTO WHERE Nombre=@NombreDepto)

INSERTINTO [MODPER].MSTEMPLEADOS(CodDepto, Nombres, Apellidos, Direccion,
Telefono, Dui, Nit, Estado)
VALUES (@CodDep, @Nom, @Ape, @Direcc, @Tel, @Du, @Ni, @Estad)
END
GO

SETANSI_NULLSON

```

```

GO
SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_MOSTRAREMPLEADOS]

AS
BEGIN

        SETNOCOUNT ON;

SELECT a.CodEmpleado, b.Nombre, a.Nombres, a.Apellidos, a.Direccion,
a.Telefono, a.Dui, a.Nit, a.Estado
FROM DBCEMENTERIO.MODPER.MSTEMPLEADOS a,
DBCEMENTERIO.MODPER.CATDEPARTAMENTO b
WHERE a.CodDepto = b.CodDepto
END
GO

SETANSI_NULLS ON
GO
SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_VISUALIZAREMPLEADOS]

AS
BEGIN

        SETNOCOUNT ON;

        SELECT CodEmpleado, Nombres, Apellidos
        FROM DBCEMENTERIO.MODPER.MSTEMPLEADOS
END
GO

SETANSI_NULLS ON
GO
SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_VISUALIZARVACACIONES]

        @Codigo int

AS
BEGIN

        SETNOCOUNT ON;

SELECT CodVacaciones, CodEmpleado, FechaInicio, FechaFin
FROM MODPER.TBLVACACIONES
WHERE CodEmpleado = @Codigo
ORDERBY CodVacaciones ASC
END
GO

SETANSI_NULLS ON
GO

```



```

SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_VERHORARIOS]

    @Codigo int

AS
BEGIN

    SETNOCOUNT ON;

SELECT CodDia, CodEmpleado, Dia, Hora
FROM MODPER.TBLHORARIO
WHERE CodEmpleado = @Codigo
ORDERBY Dia ASC
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_MODIFICARVACACION]

    @llabe int,
    @CodEmp int,
    @FechaInicio date,
    @FechaFinal date

AS
BEGIN

    SETNOCOUNT ON;

UPDATE [MODPER].TBLVACACIONES
SET FechaInicio=@FechaInicio, FechaFin=@FechaFinal
WHERE (CodEmpleado=@CodEmp) AND (CodVacaciones=@llabe)
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIER ON
GO
CREATEPROCEDURE [MODPER].[SP_MODIFICARHORARIO]

    @llabe int,
    @CodEmp int,
    @Day varchar(50),
    @Hour varchar(4)

AS
BEGIN

    SETNOCOUNT ON;

```

```

UPDATE [MODPER].TBLHORARIO
SET Dia=@Day, Hora=@Hour
WHERE (CodEmpleado=@CodEmp) AND (CodDia=@llabe)
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_INSERTARVACACION]

    @Codigo int,
    @Date1 date,
    @Date2 date

AS
BEGIN
    SETNOCOUNTON;

    INSERTINTO [MODPER].TBLVACACIONES (CodEmpleado, FechaInicio, FechaFin)
VALUES (@Codigo, @Date1, @Date2)
END
GO

SETANSI_NULLSON
GO
SETQUOTED_IDENTIFIERON
GO
CREATEPROCEDURE [MODPER].[SP_INSERTARHORARIO]

    @Codigo int,
    @Day varchar (50),
    @Hour varchar (4)

AS
BEGIN
    SETNOCOUNTON;

    INSERTINTO [MODPER].TBLHORARIO (CodEmpleado, Dia, Hora)
VALUES (@Codigo, @Day, @Hour)
END
GO

```

PROCEDIMIENTOS ALMACENADOS DEL MODULO DE COBROS Y COSTOS (MODCYC)

A continuación se detallan uno a uno los procedimientos utilizados por las clases de este módulo.

Procedimiento almacenado para anular formulas ISAM

```

USE [DBCEMENTERIO]
GO

```

```

/***** Object: StoredProcedure [MODCYC].[SP_ANULARISAM]      Script Date:
05/06/2011 10:03:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [MODCYC].[SP_ANULARISAM]

    @Isam integer,
    @Isdem integer,
    @Serie varchar(2),
    @FechaAnulacion datetime,
    @UsuarioAnulacionIsam varchar(250),
    @Motivo varchar(250)

AS
BEGIN

UPDATE [DBCENTERIO].[MODCYC].[MSTISAM]

SET     [Estado] = 'False',
        [Motivo] = @Motivo,
        [FechaHoraAnulacion] = @FechaAnulacion ,
        [UsuarioAnulacionIsam] = @UsuarioAnulacionIsam

WHERE (NumeroIsam=@Isam and Isdem=@Isdem and Serie=@Serie and Estado =
'True')

END

```

Procedimiento almacenado para buscar formulas ISAM

```

USE [DBCENTERIO]

GO

/***** Object: StoredProcedure [MODCYC].[SP_BUSCARFACTURA]      Script
Date: 05/06/2011 10:04:39 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [MODCYC].[SP_BUSCARFACTURA]

    @Isam integer,
    @Isdem integer,
    @Serie varchar(2)

AS
BEGIN

SELECT NumeroIsam, Isdem, Serie, Fecha, Total
FROM DBCENTERIO.MODCYC.MSTISAM
WHERE (NumeroIsam=@Isam and Isdem=@Isdem and Serie=@Serie and
Estado = 'True')

END

```

Procedimiento almacenado para mostrar detalle ISAM

```
USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_DETALLEISAM]      Script
Date: 05/06/2011 10:05:52 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_DETALLEISAM]

@NumeroIsam int,
@Isdem int,
@Serie varchar(2)

AS
BEGIN

        SELECT      MODCYC.TBLDETALLEISAM.Cantidad,
MODCYC.TBLDETALLEISAM.Nombre,
MODCYC.TBLDETALLEISAM.ValorUnitario,MODCYC.TBLDETALLEISAM.SubTotal
        FROM        MODCYC.MSTISAM INNER JOIN MODCYC.TBLDETALLEISAM ON
MODCYC.MSTISAM.CodIsam = MODCYC.TBLDETALLEISAM.CodIsam
        WHERE       (MODCYC.MSTISAM.NumeroIsam = @NumeroIsam) AND
(MODCYC.MSTISAM.Isdem = @Isdem) AND (MODCYC.MSTISAM.Serie = @Serie)

END
```

Procedimiento almacenado para imprimir ISAM anuladas

```
USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_IMPRIMIRISAMANULADAS]
Script Date: 05/06/2011 10:09:09 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_IMPRIMIRISAMANULADAS]
@FechaInicio date,
@FechaFinal date

AS
BEGIN

        SET NOCOUNT ON;
        SELECT NumeroIsam, Isdem, Serie, Fecha, Total FROM
DBCEMENTERIO.MODCYC.MSTISAM WHERE (Fecha BETWEEN @FechaInicio and
@FechaFinal) and Estado='False'

END
```

Procedimiento almacenado para imprimir las ISAM emitidas

```
USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_IMPRIMIRISAMNOANULADAS]
Script Date: 05/06/2011 10:10:03 *****/
```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [MODCYC].[SP_IMPRIMIRISAMNOANULADAS]

    @FechaInicio date,
    @FechaFinal date
AS
BEGIN

    SET NOCOUNT ON;

    SET NOCOUNT ON;
    SELECT NumeroIsam, Isdem, Serie, Fecha, Total FROM
DBCEMENTERIO.MODCYC.MSTISAM WHERE (Fecha BETWEEN @FechaInicio and
@FechaFinal) and Estado='True'

END

```

Procedimiento almacenado para insertar detalle ISAM

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_INSERTARDETALLEISAM]
Script Date: 05/06/2011 10:10:45 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_INSERTARDETALLEISAM]

    @NumIsa int,
    @Isdem int,
    @Serie varchar(2),
    @Cant int,
    @Type varchar(9),
    @Name varchar(250),
    @Vunitario float,
    @Stotal float
AS
BEGIN

    SET NOCOUNT ON;
    DECLARE @CodObtenido int

    SET @CodObtenido = (SELECT CodIsam FROM
DBCEMENTERIO.MODCYC.MSTISAM WHERE (NumeroIsam=@NumIsa AND Isdem= @Isdem
AND Serie=@Serie))

    INSERT INTO [MODCYC].TBLDETALLEISAM (CodIsam, Cantidad, Tipo,
Nombre, ValorUnitario, SubTotal)

```

```

VALUES (@CodObtenido, @Cant, @Type, @Name, @Vunitario, @Stotal)
END

```

Procedimiento almacenado para insertar servicios

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_INSERTARSERVICIO]      Script
Date: 05/06/2011 10:13:08 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_INSERTARSERVICIO]

    @Nombre varchar(250),
    @Descripcion varchar(350),
    @Tipo varchar(9),
    @Valor float

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO [MODCYC].MSTSERVICIO (Nombre, Descripcion, TipoServicio,
Valor)
VALUES (@Nombre, @Descripcion, @Tipo, @Valor)
END

```

Procedimiento almacenado para insertar impuestos

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_INSERTARIMPUESTO]      Script
Date: 05/06/2011 10:11:51 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_INSERTARIMPUESTO]

    @Nombre varchar(250),
    @Descripcion varchar(350),
    @Valor float

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO [MODCYC].MSTIMPUESTO (Nombre, Descripcion, Valor)
VALUES (@Nombre, @Descripcion, @Valor)
END

```

Procedimiento almacenado para agregar una ISAM

```
USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_INSERTARISAM]      Script
Date: 05/06/2011 10:12:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_INSERTARISAM]

    @NumIsa int,
    @Isdemm int,
    @Seriee varchar(2),
    @NombreClient varchar(250),
    @Datee datetime,
    @Status bit,
    @Totall float,
    @UsuarioAgregoIsam int

AS
BEGIN

    SET NOCOUNT ON;

    DECLARE @CodigoCliente int
    SET @CodigoCliente = (SELECT CodCliente FROM MODRIN.MSTCLIENTES
WHERE (Nombres + ' ' + Apellidos)=@NombreClient)
    INSERT INTO [MODCYC].MSTISAM (NumeroIsam, Isdem, Serie, CodCliente,
Fecha, Estado, Total, UsuarioCreacionIsam)
    VALUES (@NumIsa, @Isdemm, @Seriee, @CodigoCliente, @Datee, @Status,
@Totall, @UsuarioAgregoIsam)
END
```

Procedimiento almacenado para modificar impuestos

```
USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_MODIFICARIMPUESTO]
Script Date: 05/06/2011 10:13:43 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_MODIFICARIMPUESTO]

    @CodImpuesto int,
    @Nombre varchar(250),
    @Descripcion varchar(350),
    @Valor float

AS
BEGIN

    SET NOCOUNT ON;
```

```

UPDATE [MODCYC].MSTIMPUESTO
SET Nombre=@Nombre, Descripcion=@Descripcion, Valor=@Valor
WHERE CodImpuesto=@CodImpuesto
END

```

Procedimiento almacenado para modificar servicios

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_MODIFICARSERVICIO]
Script Date: 05/06/2011 10:14:34 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_MODIFICARSERVICIO]

    @CodServicio int,
    @Nombre varchar(250),
    @TipoSe varchar(9),
    @Descripcion varchar(350),
    @Valor float

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE [MODCYC].MSTSERVICIO
    SET Nombre=@Nombre, Descripcion=@Descripcion,
    TipoServicio=@TipoSe, Valor=@Valor
    WHERE CodServicio=@CodServicio
END

```

Procedimiento almacenado para mostrar impuestos

```

USE [DBCEMENTERIO]
GO
/***** Object: StoredProcedure [MODCYC].[SP_VERIMPUESTOS]      Script
Date: 05/06/2011 10:46:05 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_VERIMPUESTOS]

AS
BEGIN

    SET NOCOUNT ON;

    SELECT CodImpuesto, Nombre, Descripcion, Valor
    FROM MODCYC.MSTIMPUESTO
END

```


Procedimiento almacenado para mostrar servicios

```
USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODCYC].[SP_VERSERVICIOS]      Script
Date: 05/06/2011 10:46:40 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_VERSERVICIOS]

AS
BEGIN

    SET NOCOUNT ON;

    SELECT CodServicio, Nombre, Descripcion, TipoServicio, Valor
    FROM MODCYC.MSTSERVICIO
END
```

Procedimiento almacenado para ver clientes en la formula ISAM

```
USE [DBCEMENTERIO]
GO
/***** Object:  StoredProcedure [MODCYC].[SP_VERCLIENTES]      Script
Date: 05/06/2011 10:16:51 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [MODCYC].[SP_VERCLIENTES]

AS
BEGIN

    SET NOCOUNT ON;

    SELECT CodCliente, Nombres + ' ' + Apellidos AS nombrecompleto,
    NumeroDui, Telefono, Direccion, EstadoCivil, FechaNacimiento
    FROM DBCEMENTERIO.MODRIN.MSTCLIENTES
END
```

PROCEDIMIENTOS ALMACENADOS DEL MODULO AGENDA (MODAGA)

A continuación se detallan uno a uno los procedimientos utilizados por las clases de este módulo.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
```

```
-- Description:  Verifica si una actividad esta siendo ingresada dos veces
```

```
CREATE PROCEDURE [MODAGA].[SP_VERIFICARACTIVIDAD]
```

```
    @Nombre varchar(50)
```

```
AS  
BEGIN
```

```
    SELECT      CodActividad  
    FROM        MODAGA.MSTACTIVIDAD  
    WHERE       Nombre = @Nombre AND  
               Estado = 1
```

```
END  
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
-- Author:      MeMuX
```

```
-- Description:  Muestra los planes de trabajo
```

```
CREATE PROCEDURE [MODAGA].[SP_VERPLANES]
```

```
AS  
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    SELECT  CodPlan,  
           Nombre,  
           Descripcion,  
           Anio  
    FROM    [MODAGA].MSTPLAN
```

```
END  
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
-- Author:      MeMuX
```

```
-- Description:  Verifica si un plan esta siendo ingresado dos veces
```

```
CREATE PROCEDURE [MODAGA].[SP_VERIFICARPLAN]
```

```
    @Nombre varchar(50)
```

```
AS  
BEGIN
```

```
    SELECT      CodPlan  
    FROM        MODAGA.MSTPLAN  
    WHERE       Nombre = @Nombre
```

```

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades en base a su estado

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADES]

    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      CodActividad,
                Nombre,
                Descripcion
    FROM        MODAGA.MSTACTIVIDAD
    WHERE       Estado = @Estado
    ORDER BY   CodActividad ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Modifica el plan de trabajo en base a su codigo

CREATE PROCEDURE [MODAGA].[SP_MODIFICARPLAN]

    @Codigo int,
    @Nombre varchar(50),
    @Descripcion varchar(150),
    @Anio int

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE [MODAGA].MSTPLAN
    SET     Nombre = @Nombre,
           Descripcion = @Descripcion,
           Anio = @Anio
    WHERE  CodPlan = @Codigo

END
GO

SET ANSI_NULLS ON

```

```

GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Guarda una nueva actividad

CREATE PROCEDURE [MODAGA].[SP_GUARDARACTIVIDAD]

    @Nombre varchar(50),
    @Descripcion varchar(150)

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO    MODAGA.MSTACTIVIDAD
                  (Nombre,
                   Descripcion,
                   Estado)
    VALUES        (@Nombre,
                   @Descripcion,
                   1)

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [MODAGA].[SP_MODIFICARACTIVIDAD]

    @Codigo int,
    @Nombre varchar(50),
    @Descripcion varchar(150),
    @Motivo varchar(250),
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.MSTACTIVIDAD
    SET     Nombre = @Nombre,
           Descripcion = @Descripcion,
           Motivo = @Motivo,
           Estado = @Estado
    WHERE  CodActividad = @Codigo

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

-- Author:      MeMuX
-- Description: Guarda un nuevo plan de trabajo

CREATE PROCEDURE [MODAGA].[SP_GUARDARPLAN]

    @Nombre varchar(50),
    @Descripcion varchar(150),
    @Anio int

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO [MODAGA].MSTPLAN
        (Nombre,
        Descripcion,
        Anio)
    VALUES
        (@Nombre,
        @Descripcion,
        @Anio)

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description: Modifica una actividad del plan de trabajo

CREATE PROCEDURE [MODAGA].[SP_MODIFICARACTIVIDADPLAN]

    @CodigoPlan int,
    @CodigoActividad int,
    @Motivo varchar(250),
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.TBLDETALLES
    SET
        Motivo = @Motivo,
        Estado = @Estado
    WHERE
        CodPlan = @CodigoPlan AND
        CodActividad = @CodigoActividad

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX

```

```

-- Description:  Agrega una nueva actividad al plan de trabajo

CREATE PROCEDURE [MODAGA].[SP_GUARDARACTIVIDADPLAN]

    @CodigoPlan int,
    @CodigoActividad int

AS
BEGIN

    SET NOCOUNT ON;

    INSERT INTO    MODAGA.TBLDETALLES
                  (CodPlan,
                   CodActividad,
                   Estado)
    VALUES        (@CodigoPlan,
                   @CodigoActividad,
                   1)

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades en base a codigo de plan

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESPLANTODO]

    @Codigo int,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT        a.CodActividad,
                  a.Nombre,
                  a.Descripcion
    FROM          MODAGA.MSTACTIVIDAD a,
                  MODAGA.TBLDETALLES d
    WHERE        d.CodPlan = @Codigo AND
                  a.CodActividad = d.CodActividad AND
                  d.Estado = @Estado

    ORDER BY    a.CodActividad ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX

```

-- Description: Muestra las actividades en base a codigo de plan y estado de la actividad

```
CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESPLAN]
```

```
    @Codigo int,  
    @Estado bit
```

```
AS  
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    SELECT      a.CodActividad,  
               a.Nombre,  
               a.Descripcion  
    FROM        MODAGA.MSTACTIVIDAD a,  
               MODAGA.TBLDETALLES d  
    WHERE       d.CodPlan = @Codigo AND  
               a.CodActividad = d.CodActividad AND  
               d.Estado = @Estado AND  
               a.Estado = @Estado  
    ORDER BY   a.CodActividad ASC
```

```
END  
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
-- Author:      MeMuX
```

```
-- Description: Verifica si una actividad de un empleado ya esta  
                asignada
```

```
CREATE PROCEDURE [MODAGA].[SP_VERIFICARACTIVIDADEMPLEADO]
```

```
    @CodigoEmpleado int,  
    @CodigoActividad int,  
    @Fecha date
```

```
AS  
BEGIN
```

```
    SELECT      CodeA  
    FROM        MODAGA.TBLEMPLACT  
    WHERE       CodEmpleado = @CodigoEmpleado AND  
               CodActividad = @CodigoActividad AND  
               FechaInicio = @Fecha AND  
               Finalizado = 0 AND  
               Estado = 1
```

```
END  
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```

GO
-- Author:      MeMuX
-- Description: Muestra los empleados en base a codigo de plan

CREATE PROCEDURE [MODAGA].[SP_VEREMPLEADOSPLANTODO]

    @Codigo int,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      e.CodEmpleado,
                e.Nombres,
                e.Apellidos
    FROM        MODPER.MSTEMPLEADOS e,
                MODAGA.TBLPLEM p
    WHERE       p.CodPlan = @Codigo AND
                e.CodEmpleado = p.CodEmpleado AND
                p.Estado = @Estado

    ORDER BY   e.CodEmpleado ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description: Muestra los empleados en base a codigo de plan y estado
del empleado

CREATE PROCEDURE [MODAGA].[SP_VEREMPLEADOSPLAN]

    @Codigo int,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      e.CodEmpleado,
                e.Nombres,
                e.Apellidos
    FROM        MODPER.MSTEMPLEADOS e,
                MODAGA.TBLPLEM p
    WHERE       p.CodPlan = @Codigo AND
                e.CodEmpleado = p.CodEmpleado AND
                p.Estado = @Estado AND
                e.Estado = @Estado

    ORDER BY   e.CodEmpleado ASC

END
GO

```



```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades que estan en progreso y no han
                sido canceladas

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESPROGRESO]

    @Fecha date,
    @Finalizado bit,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      e.CodEA,
                p.Nombres,
                p.Apellidos,
                a.Nombre,
                e.FechaInicio
    FROM        MODAGA.MSTACTIVIDAD a,
                MODAGA.TBLEMPLACT e,
                MODPER.MSTEMPLEADOS p
    WHERE       e.CodEmpleado = p.CodEmpleado AND
                a.CodActividad = e.CodActividad
                AND e.Estado = @Estado AND
                e.Finalizado = @Finalizado AND
                e.FechaInicio <= @Fecha

    ORDER BY   e.FechaInicio ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades no finalizadas entre dos fechas

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESPENDIENTESFECHAS]

    @FechaInicio date,
    @FechaFin date,
    @Estado bit,
    @Finalizado bit

AS
BEGIN

    SET NOCOUNT ON;

```

```

SELECT      p.Nombres,
            p.Apellidos,
            a.Nombre,
            e.FechaInicio
FROM        MODAGA.MSTACTIVIDAD a,
            MODAGA.TBLEMPLACT e,
            MODPER.MSTEMPLEADOS p
WHERE       e.CodEmpleado = p.CodEmpleado AND
            a.CodActividad = e.CodActividad
            AND e.Estado = @Estado AND
            e.Finalizado = @Finalizado AND
            e.FechaInicio >= @FechaInicio AND
            e.FechaInicio <= @FechaFin
ORDER BY   e.FechaInicio ASC
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades de los empleados que no estan
en progreso

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESPENDIENTES]

    @Codigo int,
    @FechaInicio date,
    @Estado bit,
    @Finalizado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      e.CodEA,
            a.CodActividad,
            a.Nombre,
            a.Descripcion,
            e.FechaInicio
FROM        MODAGA.MSTACTIVIDAD a,
            MODAGA.TBLEMPLACT e
WHERE       e.CodEmpleado = @Codigo AND
            a.CodActividad = e.CodActividad AND
            e.Estado = @Estado
            AND Finalizado = @Finalizado AND
            e.FechaInicio >= @FechaInicio
ORDER BY   e.FechaInicio ASC
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

```

GO
-- Author:      MeMuX
-- Description: Muestra las actividades finalizadas entre dos fechas

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESFINALIZADAS]

    @FechaInicio date,
    @FechaFin date,
    @Estado bit,
    @Finalizado bit

AS
BEGIN

    SET NOCOUNT ON;

    SELECT      e.CodEA,
                p.Nombres,
                p.Apellidos,
                a.Nombre,
                e.FechaInicio,
                e.FechaFin
    FROM        MODAGA.MSTACTIVIDAD a,
                MODAGA.TBLEMPLACT e,
                MODPER.MSTEMPLEADOS p
    WHERE       e.CodEmpleado = p.CodEmpleado AND
                a.CodActividad = e.CodActividad AND
                e.Estado = @Estado
                AND e.Finalizado = @Finalizado AND
                e.FechaFin >= @FechaInicio AND
                e.FechaFin <= @FechaFin

    ORDER BY   e.FechaInicio ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description: Muestra las actividades de los empleados basandose en
--             su codigo y estado de la actividad y si ha finalizado o no,
--             tambien en el año en el que inician las
--             actividades

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADESEMPLEADOS]

    @Codigo int,
    @Estado bit,
    @FechaInicio date,
    @FechaFin date,
    @Finalizado bit

AS
BEGIN

```

```

        SET NOCOUNT ON;

SELECT      e.CodEA,
            e.CodEmpleado,
            a.CodActividad,
            a.Nombre,
            a.Descripcion,
            e.FechaInicio
FROM        MODAGA.MSTACTIVIDAD a,
            MODAGA.TBLEMPLACT e
WHERE       e.CodEmpleado = @Codigo AND
            a.CodActividad = e.CodActividad AND
            e.Estado = @Estado AND
            e.FechaInicio >= @FechaInicio AND
            e.FechaInicio <= @FechaFin AND
            Finalizado = @Finalizado
ORDER BY   e.FechaInicio ASC
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Muestra las actividades del día que no han sido
canceladas

CREATE PROCEDURE [MODAGA].[SP_VERACTIVIDADES DIA]

    @Fecha date,
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

SELECT      p.Nombres,
            p.Apellidos,
            a.Nombre,
            e.FechaInicio,
            e.Finalizado
FROM        MODAGA.MSTACTIVIDAD a,
            MODAGA.TBLEMPLACT e,
            MODPER.MSTEMPLEADOS p
WHERE       e.CodEmpleado = p.CodEmpleado AND
            a.CodActividad = e.CodActividad AND
            e.Estado = @Estado AND
            e.FechaInicio = @Fecha
ORDER BY   e.FechaInicio ASC
END
GO

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description: Modifica la fecha de inicio de la actividad asignada al
empleado

CREATE PROCEDURE [MODAGA].[SP_MODIFICARFECHACTIVIDAD]

    @Codigo int,
    @Fecha date,
    @Motivo varchar(250),
    @Finalizado bit

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.TBLEMPLACT
    SET     FechaInicio = @Fecha,
           Motivo = @Motivo,
           Finalizado = @Finalizado
    WHERE  CodEA = @Codigo
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description: Modifica un empleado de plan de trabajo

CREATE PROCEDURE [MODAGA].[SP_MODIFICAREMPLEADOPLAN]

    @CodigoPlan int,
    @CodigoEmpleado int,
    @Motivo varchar(250),
    @Estado bit

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.TBLPLEM
    SET     Motivo = @Motivo,
           Estado = @Estado
    WHERE  CodPlan = @CodigoPlan AND
           CodEmpleado = @CodigoEmpleado
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

```
GO
-- Author:      MeMuX
-- Description:  Agrega un nuevo empleado al plan de trabajo
```

```
CREATE PROCEDURE [MODAGA].[SP_GUARDAREMPLEADOPLAN]
```

```
    @CodigoPlan int,
    @CodigoEmpleado int
```

```
AS
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    INSERT INTO    MODAGA.TBLPLEM
                  (CodPlan,
                   CodEmpleado,
                   Estado)
    VALUES        (@CodigoPlan,
                   @CodigoEmpleado,
                   1)
```

```
END
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
-- Author:      MeMuX
-- Description:  Agrega una actividad al empleado
```

```
CREATE PROCEDURE [MODAGA].[SP_GUARDARACTIVIDADEMPLEADO]
```

```
    @CodigoEmpleado int,
    @CodigoActividad int,
    @Inicio date
```

```
AS
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    INSERT INTO    MODAGA.TBLEMPLACT
                  (CodEmpleado,
                   CodActividad,
                   FechaInicio,
                   Finalizado,
                   Estado)
    VALUES        (@CodigoEmpleado,
                   @CodigoActividad,
                   @Inicio,
                   0,
                   1)
```

```
END
GO
```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Finaliza la actividad en progreso seleccionada

CREATE PROCEDURE [MODAGA].[SP_FINALIZARACTIVIDAD]

    @Codigo int,
    @Fecha date

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.TBLEMPLACT
    SET     FechaFin = @Fecha,
           Finalizado = 1
    WHERE  CodEA = @Codigo
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:      MeMuX
-- Description:  Comprueba si el empleado tiene actividades pendientes
activas asignadas en el año seleccionado

CREATE PROCEDURE [MODAGA].[SP_EMPLEADOACTIVIDADANIO]

    @Codigo int,
    @Inicio date,
    @Fin date

AS
BEGIN

    SET NOCOUNT ON;

    SELECT     CodEmpleado
    FROM       MODAGA.TBLEMPLACT
    WHERE      CodEmpleado = @Codigo AND
           Finalizado = 0 AND
           Estado = 1 AND
           FechaInicio >= @Inicio AND
           FechaInicio <= @Fin

    ORDER BY  CodEmpleado ASC
END
GO

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Comprueba si el empleado tiene actividades pendientes
activas asignadas

CREATE PROCEDURE [MODAGA].[SP_EMPLEADOACTIVIDAD]

    @Codigo int

AS
BEGIN

    SET NOCOUNT ON;

    SELECT          CodEmpleado
    FROM            MODAGA.TBLEMPLACT
    WHERE          CodEmpleado = @Codigo AND
                  Finalizado = 0 AND
                  Estado = 1

    ORDER BY      CodEmpleado ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Cancela la actividad asignada al empleado seleccionado

CREATE PROCEDURE [MODAGA].[SP_CANCELARACTIVIDAD]

    @Codigo int,
    @Motivo varchar(250)

AS
BEGIN

    SET NOCOUNT ON;

    UPDATE MODAGA.TBLEMPLACT
    SET     Motivo = @Motivo,
           Estado = 0
    WHERE  CodeA = @Codigo

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Comprueba si una actividad pendiente y activa ya ha
sido asignada a uno o mas empleados dependiendo de año

```



```

CREATE PROCEDURE [MODAGA].[SP_ACTIVIDADEMPLEADOANIO]

    @Codigo int,
    @Inicio date,
    @Fin date

AS
BEGIN

    SET NOCOUNT ON;

    SELECT          CodActividad
    FROM            MODAGA.TBLEMPLACT
    WHERE          CodActividad = @Codigo AND
                  Finalizado = 0 AND
                  Estado = 1 AND
                  FechaInicio >= @Inicio AND
                  FechaInicio <= @Fin

    ORDER BY      CodActividad ASC

END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Author:          MeMuX
-- Description:     Comprueba si una actividad pendiente y activa ya ha
--                 sido asignada a uno o mas empleados

CREATE PROCEDURE [MODAGA].[SP_ACTIVIDADEMPLEADO]

    @Codigo int

AS
BEGIN

    SET NOCOUNT ON;

    SELECT          CodActividad
    FROM            MODAGA.TBLEMPLACT
    WHERE          CodActividad = @Codigo AND
                  Finalizado = 0 AND
                  Estado = 1

    ORDER BY      CodActividad ASC

END
GO

```

6.3 DICCIONARIO DE DATOS

MODULO DE USUARIOS (MODUSU)

A continuación se definirá el diccionario de datos correspondiente a los procedimientos inmersos en el módulo de usuario por parte de la capa de aplicación.

LOGIN DE USUARIO

LOGIN DE USUARIO = Nombre Usuario + Contraseña

ADMINISTRACIÓN DE USUARIOS

AGREGAR USUARIO = Nombre Usuario + Contraseña +
Confirmar contraseña + Rol

MODIFICAR USUARIO = Nombre de Usuario + Estado + Rol

Nombre Usuario = *Nombre del usuario del sistema*

Contraseña = *Contraseña del usuario del sistema*

Confirmar contraseña = *Misma contraseña del usuario del sistema*

Estado = *[Activo | Inactivo]*

Rol = *[Administrador/a | Secretario/a | Consultor/a]*

MODIFICAR CONTRASEÑA

MODIFICAR CONTRASEÑA = Nombre Usuario + Contraseña actual +
Nueva contraseña + Confirmar contraseña

Nombre Usuario = *Nombre del usuario del sistema*

Contraseña actual = *Contraseña actual del usuario del sistema*

Nueva contraseña = *Nueva contraseña del usuario del sistema*

Confirmar contraseña = *Misma nueva contraseña del usuario*

MODULO DE REGISTRO DE INFORMACIÓN (MODRIN)

A continuación se definirá el diccionario de datos correspondiente a los procedimientos inmersos en el módulo de registro de información por parte de la capa de aplicación.

ADMINISTRACION TESTIGOS

GUARDAR TESTIGO = NumeroPartidaDefuncion
+ Nombres
+ Apellidos
+ DUI
+Parentezco [Conyugue|Hermano|Primo|Padre
|Madre|Tio|Sobrino]

ADMINISTRAR CLIENTES

GUARDAR CLIENTE = Nombres+
Apellidos+
EstadoCivil [S|C|V|D|A]+
Direccion+
DUI+
LugarExtensionDui+
FechaNacimiento+
Telefono

MODIFICAR CLIENTE = CodCliente+
Nombres+
Apellidos+
EstadoCivil [S|C|V|D|A]+
Direccion+

DUI+
LugarExtensionDui+
FechaNacimiento+
Telefono

GUARDAR INHUMACION = Codigolsam+
PartidaDefuncion+
Propiedad+
FechaInhumacion+
CausasFallecimiento+
DireccionFallecimiento+
LugarFallecimiento

GUARDAR EXHUMACIÓN = CodExhumacion+
CodInhumacion+
FechaExhumacion+
Causas+
Observaciones

CREAR PARTIDA DEFUNCION = NumPartida+
NumIsam+
NombreSolicitante+
NombreFallecido+
PadreFallecido+
MadreFallecido+
Oficio+

EstadoCivil [S|C|V|D|A]+
Edad+
CausaMuerte+
Genero [M|F]
Direccion+
TipoFallecimiento[Casa|Hospital|Med. Legal]+
Hora+
Fecha+
Medico

VENDER PROPIEDAD = DatosComprador+
DatosPropiedad+
{Puesto}+
{Sepultura}

MODULO DE PERSONAL (MODPER)

A continuación se definirá el diccionario de datos correspondiente a los procedimientos inmersos en el módulo de personal por parte de la capa de aplicación.

Agregar Departamento

Agregar departamento = Nombre
Nombre = Nombre del departamento

Modificar departamento

Modificar departamento = Nombre
Nombre = Nombre del departamento

Agregar Empleado

Agregar Empleado	=	CodDepto+ Nombres+ Apellidos+ Direccion+ Telefono+ DUI+ NIT+ Estado
CodDepto	=	Codigo de un departamento
Nombres	=	Primer Nombre+ Segundo Nombre
Apellidos	=	Primer Apellido+ Segundo Apellido
Direccion	=	Departamento+Colonia+Casa
Telefono	=	Numero Local
DUI	=	Documento único de identidad
NIT	=	Numero de NIT
Estado	=	Activo

Modificar Empleado

Modificar Empleado	=	CodDepto+ Nombres+ Apellidos+ Direccion+ Telefono+ DUI+ NIT+ Estado
CodDepto	=	Codigo de un departamento
Nombres	=	Primer Nombre+

Apellidos	=	Segundo Nombre Primer Apellido+ Segundo Apellido
Direccion	=	Departamento+Colonia+Casa
Telefono	=	Numero Local
DUI	=	Documento único de identidad
NIT	=	Numeo de NIT
Estado	=	[Activo Inactivo]

Modificar Contratista

Modificar Contratista	=	Nombres+ Apellidos+ Direccion+ Telefono+ DUI+ NIT+ Estado
Nombres	=	Primer Nombre+ Segundo Nombre
Apellidos	=	Primer Apellido+ Segundo Apellido
Direccion	=	Departamento+Colonia+Casa
Telefono	=	Numero Local
DUI	=	Documento único de identidad
NIT	=	Numeo de NIT
Estado	=	[Activo Inactivo]

Agregar Horario

Agregar Horario	=	CodEmpleado+ Dia+ Hora
CodEmpleado	=	Código de un empleado
Dia	=	Día de la semana

Hora = Numero de horas laborales

Modificar Horario

Modificar Horario = CodEmpleado+
Dia+
Hora

CodEmpleado = Código de un empleado
Dia = Día de la semana
Hora = Numero de horas laborales

Agregar Vacaciones

Agregar Vacaciones = CodEmpleado+
FechaInicio+
FechaFin

CodEmpleado = Código de un empleado
FechaInicio = Inicio de vacaciones
FechaFinal = Fecha de fin de vacaciones.

Modificar Vacaciones

Modificar Vacaciones = CodEmpleado+
FechaInicio+
FechaFin

CodEmpleado = Código de un empleado
FechaInicio = Inicio de vacaciones
FechaFinal = Fecha de fin de vacaciones.

MODULO DE COBROS Y COSTOS (MODCYC)

A continuación se definirá el diccionario de datos correspondiente a los procedimientos inmersos en el módulo de cobros y costos por parte de la capa de aplicación.

Agregar Factura ISAM

Agregar ISAM = NumeroISam+

Isdem+
Serie+
CodCliente+
Fecha+
Estado+
Total+

Numerolsam	=	Numero de la factura
Isdem	=	Numero de Isdel
Serie	=	Serie de la factura
CodCliente	=	Código de un cliente
Fecha	=	Fecha de emisión
Estado	=	[Activo Inactivo]
Total	=	Sumatoria de costos

Anular Factura ISAM

Agregar ISAM	=	NumerolSam+ Isdem+ Serie+ CodCliente+ Fecha+ Estado+ Total+
Numerolsam	=	Numero de la factura
Isdem	=	Numero de Isdel
Serie	=	Serie de la factura
CodCliente	=	Código de un cliente
Fecha	=	Fecha de emisión
Estado	=	Inactivo
Total	=	Sumatoria de costos

Agregar Impuesto

Agregar Impuesto	=	Nombre+ Descripcion+ Valor
------------------	---	----------------------------------

Nombre	=	Nombre del impuesto
Descripcion	=	Descripcion del impuesto
Valor	=	Valor del impuesto

Modificar Impuesto

Modificar Impuesto	=	Nombre+ Descripcion+ Valor
--------------------	---	----------------------------------

Nombre	=	Nombre del impuesto
Descripcion	=	Descripcion del impuesto
Valor	=	Valor del impuesto

Agregar Servicio

Agregar Servicio	=	Nombre+ Descripcion+ Valor
------------------	---	----------------------------------

Nombre	=	Nombre del impuesto
Descripcion	=	Descripcion del impuesto
Valor	=	Valor del impuesto

Modificar Servicio

Modificar Servicio	=	Nombre+ Descripcion+ Valor
--------------------	---	----------------------------------

Nombre	=	Nombre del impuesto
Descripcion	=	Descripcion del impuesto
Valor	=	Valor del impuesto

Agregar Detalle Isam

Agregar Detalle ISam	=	CodIsam+ CodServicio+ CodImpuesto SubTotal
----------------------	---	---

CodIsam	=	Código de la factura
CodServicio	=	Código del servicio
CodImpuesto	=	Código del Impuesto
SubTotal	=	Sumatoria de valores

MODULO DE AGENDA (MODAGA)

A continuación se definirá el diccionario de datos correspondiente a los procedimientos inmersos en el módulo de agenda por parte de la capa de aplicación.

PLANES DE TRABAJO

GUARDAR PLAN DE TRABAJO = Nombre plan + Año + Descripción

MODIFICAR PLAN DE TRABAJO = Nombre plan + Año + Descripción

Nombre Plan = *Nombre del plan de trabajo*

Año = *Año del plan de trabajo*

Descripción = *Descripción del plan de trabajo*

ACTIVIDADES

GUARDAR ACTIVIDAD = Nombre actividad + Descripción

MODIFICAR ACTIVIDAD = Nombre actividad + Descripción

Nombre actividad = *Nombre de la actividad*

Descripción = *Descripción de la actividad*

EMPLEADOS DEL PLAN DE TRABAJO

AGREGAR EMPLEADOS AL PLAN = Código plan + Código empleado

Código plan = *Código del plan de trabajo*

Código empleado = *Código del empleado*

ACTIVIDADES DEL PLAN DE TRABAJO

AGREGAR ACTIVIDADES AL PLAN = Código plan + Código actividad

Código plan = *Código del plan de trabajo*

Código actividad = *Código de la actividad*

ADMINISTRACIÓN DE ACTIVIDADES DEL EMPLEADO

ASIGNAR ACTIVIDAD AL EMPLEADO = Código empleado + Código actividad +
Fecha Inicio

Código empleado = *Código del empleado*

Código actividad = *Código de la actividad*

Fecha Inicio = *Fecha de inicio de la actividad*

CONTROL DE ACTIVIDADES

MOSTRAR ACTIVIDADES DEL DÍA	=	Fecha Inicio
MOSTRAR ACTIVIDADES PROGRESO	=	Nada
MOSTRAR ACTIVIDADES PENDIENTES	=	Fecha Inicio + Fecha Fin
MOSTRAR ACTIVIDADES REALIZADAS	=	Fecha Inicio + Fecha Fin
<i>Fecha Inicio</i>	=	<i>Fecha de inicio del intervalo de actividades</i>
<i>Fecha Fin</i>	=	<i>Fecha de finalización del intervalo de actividades</i>
<i>Nada</i>	=	<i>Significa que no se necesitan datos</i>

ELIMINACIÓN Y CANCELACIÓN DE ACTIVIDADES/EMPLEADOS

ELIMINAR ACTIVIDAD	=	Código actividad + Motivo
ELIMINAR ACTIVIDAD DEL PLAN	=	Código plan + Código actividad + Motivo
ELIMINAR EMPLEADO DEL PLAN	=	Código plan + Código empleado + Motivo
CANCELAR ACTIVIDAD	=	Código empleado + Código actividad
<i>Código actividad</i>	=	<i>Código de la actividad</i>
<i>Código plan</i>	=	<i>Código del plan de trabajo</i>
<i>Código empleado</i>	=	<i>Código del empleado</i>
<i>Motivo</i>	=	<i>Motivo de la eliminación o cancelación</i>

FINALIZACIÓN DE ACTIVIDADES

FINALIZAR ACTIVIDAD	=	Código actividad
<i>Código actividad</i>	=	<i>Código de la actividad</i>

CAPITULO VII.
CONCLUSIONES Y
RECOMENDACIONES

7.1 CONCLUSIONES

El Cementerio Santa Isabel es una de las entidades más importantes en el municipio de Santa Ana, en un apoyo a dicha institución se tomó a bien desarrollar un Sistema de Información para el manejo de información interna, proveyendo una herramienta de trabajo que permita ayudar a las personas que laboran en dicho lugar a realizar sus labores de una forma más eficiente en sus actividades diarias.

A partir del trabajo de grado realizado en el Cementerio Santa Isabel se concluye que se automatizaron procesos que antes se realizaban de forma manual y que involucraban demoras de tiempo y enormes volúmenes de material como por ejemplo papel, folios y libros entre otros, siendo estos en cierta medida ineficientes ante la demanda que se tiene ahora en día de los servicios que dicha institución brinda a la población.

A demás se permitió llevar un mejor control y manejo de la información comparado a los métodos antes mencionados, ya que ahora estará en una base de datos la cual podrá ser asegurada mediante respaldo a las bases de datos y no como en libros sujetos a deterioro tal y como ocurría antes de la llegada del sistema de información.

También se puede decir que el proyecto realizado permitió optimizar algunos procesos de trabajo mediante el uso de recursos informáticos, aprovechando así las bondades que estos brindan tanto en velocidad, seguridad y disponibilidad de la información para el usuario al momento que este la desee.

Finalmente se da por satisfecho el trabajo realizado en el desarrollo del presente proyecto, contando con el visto bueno de las personas que laboran en el Cementerio Santa Isabel, quedando claro que se dará un mejor servicio a la población mediante el buen uso del mismo.

7.2 RECOMENDACIONES

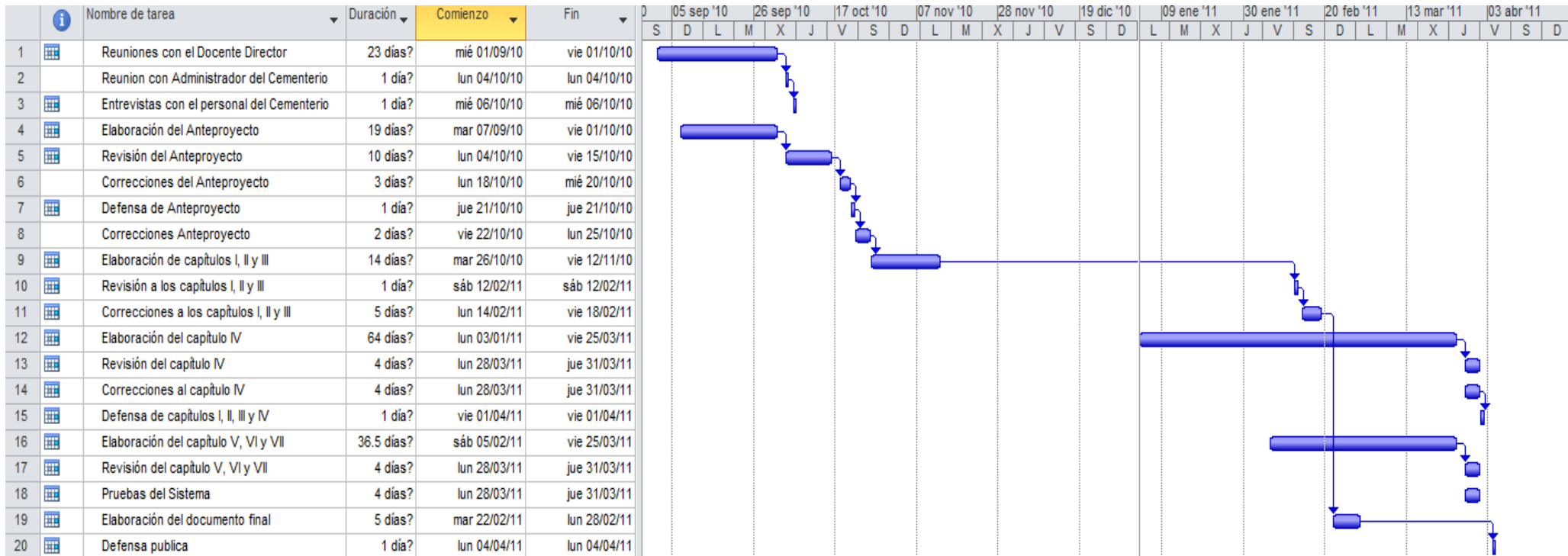
- Proteger con un UPS cada estación de trabajo.
- Contar con equipos que reúnan los requerimientos mínimos en base a hardware y software para soportar el gestor de base de datos.
- Contar con software antivirus en cada equipo de cómputo.
- Instalar Microsoft.NET Framework en los equipos que harán uso del sistema, esto para que sea posible la instalación del mismo.
- Instalar el gestor de base de datos SQL Server en una sesión de usuario con privilegios de administrador.
- Activar en todas las computadoras que tengan instalado el sistema un protector de pantalla con petición de contraseña a desplegarse en un lapso estimado de 10 minutos de inactividad por motivos de privacidad para el mismo, de tal forma que se evite exponer información importante a la vista de personas no autorizadas.
- El administrador del sistema tiene que ser una persona con conocimientos en Bases de datos SQL Server y aplicaciones .Net
- Establecer una política de mantenimiento y respaldos a la Base de Datos.
- Mantener en buen estado el funcionamiento de la red local de cables de red y dispositivos de red utilizados en la red local.
- Proteger el CD de instalación entregado al cementerio por parte del grupo de tesis.
- Mantener una dirección IP estática en el equipo que contendrá el gestor de base de datos.

7.3 BIBLIOGRAFÍA

- Kendall & Kendall, Análisis y diseño de sistema, sexta edición, Pearson Education, México 1995.
- Joseph Schmuller, Aprendiendo UML en 24 horas.
- Ingeniería Económica 5ta edición - Leland T. Blank y Anthony J. Tarquin

7.4 ANEXOS

7.4.1 Cronograma de actividades



7.4.2 Presupuesto general

PRESUPUESTO GENERAL PARA IMPLEMENTACION DEL PROYECTO DEL SISTEMA DE INFORMACION EN EL CEMENTERIO DE SANTA ANA

No.	RUBRO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
1.-	ESTACIONES DE TRABAJO	2	\$ 848.00	\$ 1,696.00
2.-	CABLEADO ESTRUCTURADO	*	\$ 278.00	\$ 278.00
3.-	UPS (700 VA)	3	\$ 112.50	\$ 450.00
4.-	IMPRESOR FX-890	2	\$ 450.00	\$ 900.00
5.-	POLARIZACION	5	\$ 62.30	\$ 311.50
6.-	SwitchLinksys	1	\$ 29.99	\$ 29.99
7.-	DESARROLLO	3 PER.	\$ 1,200.00	\$ 1,200.00
			TOTAL	\$ 4,865.49

NOTA: DEBIDO A QUE LA ALCALDIA MUNICIPAL PRETENDE PAGAR UN ENLACE DEDICADO DEL CEMENTERIO HACIA SUS SERVIDORES SE REDUCE EL PRESUPUESTO Y SE UTILIZARIA TEMPORALMENTE UNA ESTACION DE TRABAJO COMO SERVIDOR

7.4.3 Presupuesto de equipos informáticos

PRESUPUESTO DE EQUIPOS INFORMÁTICOS PARA IMPLEMENTACION DEL PROYECTO DEL SISTEMA DE INFORMACION EN EL CEMENTERIO DE SANTA ANA

No	EQUIPO	CANTIDAD	PRECIO	TOTAL
1.-	ESTACIONES DE TRABAJO	2	\$ 848.00	\$ 1,696.00
2.-	CABLEADO ESTRUCTURADO	*	\$ 278.00	\$ 278.00
3.-	UPS (700 VA)	3	\$ 112.50	\$ 450.00
4.-	IMPRESOR FX-890	2	\$ 450.00	\$ 900.00
5.-	POLARIZACION	5	\$ 62.30	\$ 311.50
			TOTAL	\$ 3,665.49

7.4.4 Composición de equipos informáticos

COMPOSICION DE EQUIPOS INFORMÁTICOS NECESARIOS PARA LA IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN EN EL CEMENTE RIO MUNICIPAL SANTA ISABEL DE SANTA ANA

4 ESTACIONES DE TRABAJO: VOSTRO 230 ST



No	 COMPONENTES	DESCRIPCION	PRECIO
1.-	PROCESADOR	Intel® Pentium® E5400 (2MB Caché, 2.70GHz, 800MHz FSB)	\$848.00
2.-	SISTEMA OPERATIVO	Windows® 7 Profesional de 32 bit con Medio en Español	
3.-	MEMORIA	Memoria de 2GB Dual Channel DDR3 SDRAM 1333MHz - 2DIMMs	
4.-	MONITOR	Monitor Dell E1910H - Plano y Pantalla Ancha de 18.5 pulgadas	
5.-	TARJETA DE VIDEO	Integrated Video, Intel® GMA X4500	
6.-	DISCO DURO	Disco Duro de 250GB Serial ATA (7200RPM) con DataBurst Caché™	
7.-	DISPOSITIVO ÓPTICO	Single Drive: 16X (DVD+/-RW) Burner Drive	
8.-	TECLADO DELL	Dell Multimedia Teclado - Español	
9.-	MOUSE DELL	Dell Laser Mouse - Black Gloss	
10.-	LECTOR DE MEMORIA	Lector de Tarjeta de Medios Dell 8-in-1	
11.-	BOCINAS DELL	Bocinas Dell AX210 Estéreo USB	
12.-	MODEM	56K PCI Data Fax Modem	
13.-	TARJETA DE RED	Intergrated PCIE 10/100/1000	
14.-	GARANTÍA	Un Año de garantía limitada. Servicio en el sitio con respuesta el siguiente día laborable.	

7.4.5 Presupuesto del software

PRESUPUESTO DE SOFTWARE PARA LA IMPLEMENTACION DEL PROYECTO DEL SISTEMA DE INFORMACION EN EL CEMENTERIO DE SANTA ANA

No	LICENCIAS	CANTIDAD	PRECIO	TOTAL
1.-	Microsoft Windows 7 Home Basic-Español	4	\$ -	\$ -
2.-	Microsoft Office Home and Business 2010 - Español	1	\$ -	\$ -
3.-	Trend Micro Worry-Free Business Security Services, en español - 15 Meses	4	\$ -	\$ -
4.-	Microsoft Forefront Client Security (3 años)	1	\$ 152.64	\$ 152.64
			TOTAL	\$ 152.64

Nota: Las licencias que no tienen precio están incluidas en el precio de los equipos informáticos

7.4.6 Presupuesto de cableado de red

PRESUPUESTO DE CABLEADO DE RED PARA LA IMPLEMENTACION DEL PROYECTO DEL SISTEMA DE INFORMACION EN EL CEMENTERIO DE SANTA ANA

No	EQUIPO	CANTIDAD (MTS/UNIDADES)	PRECIO	TOTAL
1.-	CABLE UTP CAT-6E	100	\$ 1.00	\$ 100.00
2.-	CONECTOR RJ-45	25	\$ 0.26	\$ 6.50
3.-	CANAleta PARA CABLE UTP	50	\$ 2.50	\$ 125.00
4.-	SET CAJA DE CONECTOR RJ45	5	\$ 8.00	\$ 40.00
5.-	PROTECTORES DE RJ-45	25	\$ 0.26	\$ 6.50
			TOTAL	\$ 278.00

7.4.7 Presupuesto de polarización de la red eléctrica

PRESUPUESTO DE POLARIZACIÓN DE LA RED ELECTRICA PARA LA IMPLEMENTACION DEL PROYECTO DEL SISTEMA DE INFORMACION EN EL CEMENTERIO DE SANTA ANA

No	EQUIPO	CANTIDAD (MTS/UNIDADES)	PRECIO	TOTAL
1.-	CABLE CONDUCTOR TW CALIBRE 12	25	\$ 1.50	\$ 37.50
2.-	BARRA DE COBRE	5	\$ 40.00	\$ 200.00
3.-	SET DE TOMACORRIENTES	5	\$ 12.00	\$ 60.00
4.-	CINTA AISLANTE	1	\$ 1.50	\$ 1.50
5.-	CANAleta	5	\$ 2.50	\$ 12.50
			TOTAL	\$ 311.50

7.4.8 Carta de solicitud del sistema de información de parte del administrador del cementerio Santa Isabel

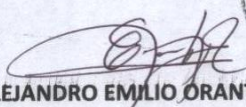
Santa Ana, 19 de Agosto de 2010

ING. RAUL MARTINEZ BERMUDEZ
JEFE DEPARTAMENTO DE INGENIERIA
Y ARQUITECTURA, FMOcc.- UES
PRESENTE.

Por este medio se hace constar la necesidad de un Sistema de Información y Administración de datos en el Cementerio Santa Isabel de Santa Ana, dado que actualmente se trabaja de forma manual y con métodos obsoletos que dificultan la adecuada administración de los datos y una atención ágil y eficaz al público

Atentamente




Sr. ALEJANDRO EMILIO ORANTES HERNANDEZ

ADMINISTRADOR CEMENTERIO GENERAL SANTA ISABEL

7.4.9 Entrevista realizada al personal administrativo del cementerio Santa Isabel y jefe del departamento de informática de Alcaldía Municipal

1. ¿Cuánto ha aumentado el número de casos atendidos en el cementerio en los últimos 5 años?
2. Describa paso a paso cada uno de los procesos que se realizan en el cementerio (administrativos y de atención al público).
3. ¿Cuenta la alcaldía con servidores?
4. ¿Qué sistema operativo utilizan los servidores?
5. ¿Es posible alojar servicios en los servidores?
6. ¿Poseen licencias de Windows disponibles?
7. ¿Utilizan Active Directory?
8. ¿Cuánto tiempo le toma en promedio brindar un servicio a los usuarios del cementerio?
9. ¿Considera que a usted como empleado se le facilitaría su trabajo al utilizar el sistema de información?
10. ¿Tiene conocimiento si en algún cementerio nacional ha implementado satisfactoriamente algún sistema de información?
11. ¿Qué problemas les genera la utilización de los procesos tal como los realizan actualmente?
12. ¿Qué inconvenientes les genera como institución en su trabajo mantener la información en libros?
13. ¿Cómo se encuentra estructurado el cementerio?
14. ¿Qué informes se generan para entregar a la alcaldía municipal?
15. ¿Cuáles son los servicios que brinda el cementerio al público en general?