Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Modelo de productividad de empleados con actividad laboral en la computadora mediante el apoyo de un agente

---

Proyecto de Tesina para la Titulación de la Licenciatura en Ciencias de la Computación

ALUMNO
Emmanuel Cabrera Aldana

ASESOR
Dr. J. Bernardo Parra Victorino

Otoño 2014

Agradecimientos

A Dios, mi esposa, mi familia, mis maestros,
mis amigos, por insistir y creer en mí.

Contenido

TITULO DEL PROYECTO DE TESINA.

Modelo de productividad de empleados con actividad laboral en la computadora mediante el apoyo de un agente.

INTRODUCCION.

En un área administrativa de Volkswagen de México, el trabajo mediante el uso de una computadora ocupa el 90% del tiempo laboral de los empleados. Sin embargo, el tiempo laboral real es menor a las 9 horas diarias establecidas para esta actividad.
Esto se debe a que diariamente se requiere tiempo para diversas actividades que incluyen la comida, el uso de sanitarios, el consumo de *snacks*, charlas, tomar un café, fumar un cigarro, llamadas telefónicas, actividades personales y ocio. Lo anterior nos lleva a la necesidad de que los empleados tengan agendas manejadas por un asistente virtual, denominado agente, que lleve un registro de las actividades del empleado, sus tiempos, temas pendientes y terminados, el tiempo que ha permanecido sin realizar actividades mediante la computadora, etc. con el fin de conocer y planificar mejor el tiempo empleado en dichas actividades diarias.

Se contará con una aplicación que apoyará la gestión de la agenda laboral y personal, facilitará tener un mejor control de los pendientes del trabajador y llevará un registro de las actividades realizadas con sus tiempos correspondientes. Esta aplicación será la encargada de enviar recordatorios programables al trabajador mediante el agente supervisor del tiempo dedicado a cada una de las actividades de las agendas. Todo esto permitirá hacer más eficiente el trabajo, pues al analizar posteriormente las estadísticas proporcionadas por el agente, será posible realizar una planeación real de todas las actividades diarias y facilitará al empleado llevar un control de sus temas pendientes sin olvidar terminarlos.

DESARROLLO

Antecedentes

Actualmente existen diversas herramientas de software que permiten llevar un control de las actividades, citas, tareas, proyectos, de las cuáles se describen brevemente dos de las más importantes: Microsoft© Outlook© 2010 y Microsoft© Project© 2010.

Microsoft© Outlook© es una herramienta de software creada para administrar de una manera amigable el correo electrónico, calendario, contactos, notas y otros tantos datos del usuario con múltiples herramientas y opciones que lo hacen el software de elección empresarial por excelencia.
Al ser parte de una *suite*, el software es robusto, funcional, y sin embargo ineficiente para el problema que se presenta. El usuario necesita cronometrar sus tiempos, Outlook© no puede hacerlo.
Se requiere también independencia de la red y portabilidad. Outlook© no las tiene.
Es un programa muy general para múltiples usuarios y lo que se requiere es una aplicación *ad hoc,* pues los propósitos son muy específicos[1].

"Microsoft© Project© Professional 2010 ofrece una forma potente y visualmente mejorada de administrar una amplia gama de proyectos y de programas eficazmente. Mediante una experiencia novedosa e intuitiva, esta solución proporciona las herramientas de planificación, administración y colaboración empresarial, de personas y de equipos necesarias para cumplir con los plazos de entrega cruciales o elegir los recursos adecuados para un equipo, entre otros objetivos.

Gracias a la nueva y mejorada vista de calendario, tendrá una visión más clara de las tareas, las fechas clave y las fases de un proyecto o programa. Con las paletas de colores y los efectos de texto ampliados logrará una visibilidad eficaz del calendario para ver y compartir las fechas clave y los plazos de entrega".[2]

Este software proporciona herramientas que ayudan a controlar perfectamente un proyecto, pero en nuestro caso, sólo necesitamos registrar los tiempos empleados en cada actividad (laboral y personal incluyendo el ocio) de una manera simple, sencilla y que permita tomar decisiones para una mejor planeación de los mismos. Necesitamos además tener una definición del perfil de usuario de acuerdo a sus preferencias en las actividades realizadas diariamente para poder sugerir de manera automática en qué emplear el tiempo entre actividades.

Hay además muchas agendas y programas en Internet que podrían servir para cubrir parte de nuestros objetivos, sin embargo, las características que requerimos

del software se encuentran distribuidas entre distintos programas en la red mundial de información, y las necesitamos en una sola aplicación.

Hay incluso programas que registran nuestras actividades en la PC, pero no es suficiente, pues no podemos tomar decisiones mediante aquéllos, ya que sólo registran, no proporcionan estadísticas o ayuda alguna para decidir en qué emplearemos el tiempo de manera adecuada.

Es por eso que la creación de una aplicación *ad hoc*, permitirá al usuario satisfacer las necesidades de administración, control y seguimiento de sus actividades laborales y personales diarias de una manera amigable, sencilla, útil y sobre todo sin excesos ni faltantes.

El resultado esperado es una aplicación que lleve un registro de actividades, tiempos y preferencias que permitan al usuario tomar decisiones al momento de planear sus actividades pendientes mediante el conocimiento de su manejo de tiempo, apoyado por el agente que supervisa actividades y el uso de estadísticas reales. Esta aplicación puede proporcionar también la información que un jefe podría utilizar para mejorar la productividad de sus empleados e incluso con fines de evaluación.

Fundamentos teóricos

Se realizará una investigación de campo con el fin de obtener la información necesaria para planear la manera más adecuada del manejo de los datos por la aplicación, el diseño de la interfaz, el almacenamiento de datos, las capacidades y limitantes de hardware, el software en la empresa, la cantidad de usuarios y sus capacidades y/o limitaciones, etc.
Esta investigación deberá ayudar también a definir qué datos y actividades son relevantes de registro y los resultados que deben mostrarse y almacenarse.

Se realizó una investigación documental sencilla que permitió llevar a cabo de una manera más efectiva el modelo de desarrollo de software elegido, que es el de Modelo de Espiral[3,4,5].
Este modelo es un modelo evolutivo (modelos cuyas principales ventajas son que se entregan versiones del sistema al usuario, se recibe su retroalimentación y con base en ésta, se reajustan el diseño y los objetivos por ciclos) y cuyas principales características son:
-Se desarrolla en ciclos
-En cada ciclo se define el objetivo, se analizan los riesgos, se desarrollan y verifican las soluciones obtenidas, y se revisan los resultados para la planificación del siguiente ciclo.
-Se enfoca en el manejo de riesgos, orientación al cliente y desarrollo iterativo.

Las ventajas del modelo son:
-Resolución temprana de riesgos
-Definición de su arquitectura en fases iniciales
-Basado en un proceso continuo de verificación de calidad
-Ideal para productos con un nivel alto de inestabilidad de los requerimientos.

Se ha elegido este modelo principalmente por la facilidad que se tiene de entregar a los usuarios las versiones de la aplicación para ser evaluadas y recibir retroalimentación para el reajuste. Además de que por la naturaleza del problema y de la empresa, es necesario evaluar constantemente la facilidad de uso de la aplicación, el uso de los datos correctos y completos, y la evaluación de los resultados obtenidos.

Agentes

Podemos definir al agente[6] inteligente como una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere.

Todos los agentes inteligentes son programas, pero no todos los programas que realizan búsquedas son agentes inteligentes. Los agentes en sí mismos pueden ser considerados como entidades individuales (partes de programa que tienen control sobre sus propias vidas y movimientos). Continuamente están realizando procesos que les indican qué hacer y cómo. Se comunican con otros agentes para resolver de forma adecuada su trabajo.

De acuerdo con el punto de vista de la inteligencia artificial un agente posee las siguientes propiedades: autonomía, sociabilidad, capacidad de reacción, iniciativa, benevolencia y racionalidad (Wooldridge y Jennings**,** 1995).

No es necesario que un agente dedicado a la recuperación de información posea todas las propiedades que se han citado, pero sí las que a continuación se describen:

*Autonomí*a: actuar sin ningún tipo de intervención humana directa, y tener control sobre sus propios actos.

Sociabilidad: comunicarse por medio de un lenguaje común con otros agentes, e incluso con los humanos.

Capacidad de reacción: percibir su entorno, y reaccionar para adaptarse a él.

Iniciativa: emprender las acciones para resolver un problema.

Los agentes de información están diseñados específicamente para procesar consultas, y poseen al menos uno de los siguientes elementos: capacidad de proceso, conocimiento del entorno donde se mueven e información de un dominio.

Existen diferentes tipos de agentes: cooperativos, móviles, interfaces inteligentes de usuario, agentes de búsqueda inteligentes, agentes de consulta, de bases de datos, de consulta de bases de datos, de usuario.

Para el desarrollo de esta aplicación, se utilizará una aproximación a un agente orientado a usuario. Los agentes orientados a usuario están asociados a una persona en concreto. A diferencia de los agentes de consulta, que se generan cada vez que tienen que realizar una consulta para un individuo, los agentes de usuario siempre están activos, buscando información y suministrándosela a su creador.

Los agentes de usuario son más efectivos cuando se trata de recuperar información que es relativamente constante, pero cuyas fuentes son dinámicas. Los agentes de usuario pueden ser utilizados para recuperar información de bases de datos, de revistas electrónicas, o incluso de los mensajes e-mail. Su objetivo es disminuir el trabajo necesario en la recuperación de la información. En definitiva, hacer la vida de «su propietario» más fácil.

El agente que se incluirá en la aplicación contará con las características mencionadas previamente: autonomía, sociabilidad, capacidad de reacción e iniciativa. Sin embargo, no tendrá capacidad de aprendizaje pues basará su comportamiento únicamente en promedios de uso, tiempos y medias.

Una vez realizadas las investigaciones, de campo y documental, se siguieron los pasos del Modelo de Espiral para el desarrollo de la aplicación.

Durante el diseño será necesario definir interfaces que sean lo más intuitivas y actuales posible para facilitar al usuario el empleo de esta aplicación.
Deberá también definirse cuál será la manera más adecuada de manejar las bases de datos por las limitantes en cuanto al software, que se tiene en el departamento de aplicación.
Se realizarán, de acuerdo al modelo elegido, prototipos que deberán ser evaluados por los usuarios al término de cada fase, pues es necesario reevaluar constantemente y en situaciones reales la utilidad de algunos campos y sobre todo, la facilidad de uso de la aplicación.

Durante estas evaluaciones, se harán anotaciones con los comentarios de los usuarios, y si es posible se definirá la factibilidad de los cambios solicitados, así como las opciones que podría haber para llevar a cabo su petición.
En cada prototipo se agregarán, modificarán y eliminarán las funciones de acuerdo a los requerimientos de los usuarios teniendo como límite el tercer prototipo, pues por el tiempo no será posible continuar con las modificaciones.

Cada prototipo será funcional hasta donde se haya avanzado en el momento de la evaluación.

Objetivo General

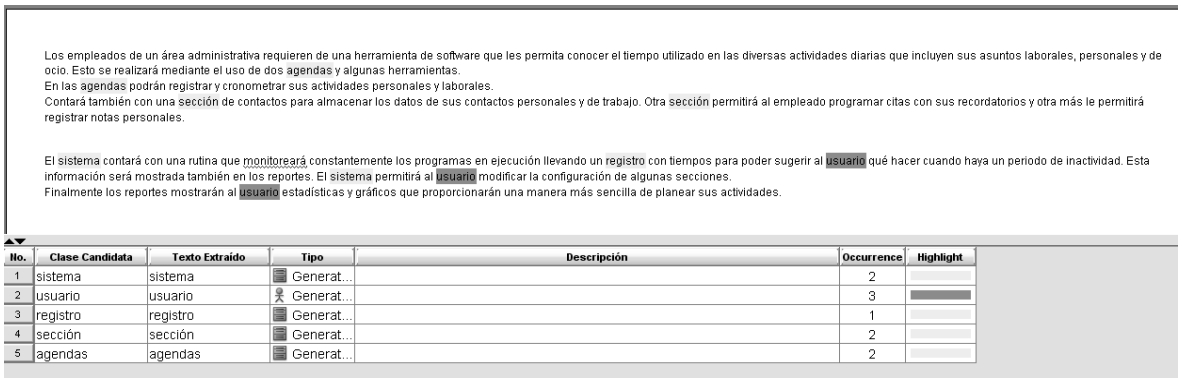
Los empleados que desarrollan sus actividades laborales de manera preponderante mediante el uso de la computadora, requieren de un asistente que les apoye en el control del desempeño de sus actividades. Por lo anterior se propone como Objetivo:
Desarrollar una aplicación que permita asistir a los empleados que realizan sus actividades laborales mediante una computadora, administrando sus agendas laboral y personal, supervisando el desarrollo adecuado de las agendas mediante un agente; así como midiendo los tiempos empleados en las diferentes actividades del empleado.

Objetivos Específicos

• Desarrollar una aplicación que permita registrar las actividades que realiza el trabajador así como el tiempo empleado en cada aplicación y/o actividad.
• Registrar las agendas personal y laboral.
• Desarrollar un agente que supervise las actividades que se han realizado y que en un tiempo determinado le proporcione estadísticas de las actividades realizadas con el objetivo de facilitar la planeación de actividades pendientes.
• Que el agente envíe un mensaje al trabajador cuando éste no realice actividades en la computadora en un tiempo programable.
• Desplegar estadísticas para la toma de decisiones.
• Así mismo la aplicación tendrá diferentes secciones como Citas, Contactos y Notas, que faciliten al trabajador la organización de sus actividades, teniendo en un solo lugar la información básica que requiera.

ANALISIS Y DISEÑO DEL SISTEMA

Análisis Textual



Los empleados de un área administrativa requieren de una herramienta de software que les permita conocer el tiempo utilizado en las diversas actividades diarias que incluyen sus asuntos laborales, personales y de ocio. Esto se realizará mediante el uso de dos agendas y algunas herramientas.
En las agendas podrán registrar y cronometrar sus actividades personales y laborales.
Contará también con una sección de contactos para almacenar los datos de sus contactos personales y de trabajo. Otra sección permitirá al empleado programar citas con sus recordatorios y otra más le permitirá registrar notas personales.

El sistema contará con una rutina que monitoreará constantemente los programas en ejecución llevando un registro con tiempos para poder sugerir al usuario qué hacer cuando haya un periodo de inactividad. Esta información será mostrada también en los reportes. El sistema permitirá al usuario modificar la configuración de algunas secciones.
Finalmente los reportes mostrarán al usuario estadísticas y gráficos que proporcionarán una manera más sencilla de planear sus actividades.

No.	Clase Candidata	Texto Extraido	Tipo	Descripción	Occurrence	Highlight
1	sistema	sistema	Generat...		2	
2	usuario	usuario	Generat...		3	
3	registro	registro	Generat...		1	
4	sección	sección	Generat...		2	
5	agendas	agendas	Generat...		2	

Figura 1. Esta imagen muestra el análisis textual del problema a resolver, permitiendo identificar las posibles clases involucradas, así como los actores.

Diagrama General de Casos de Uso

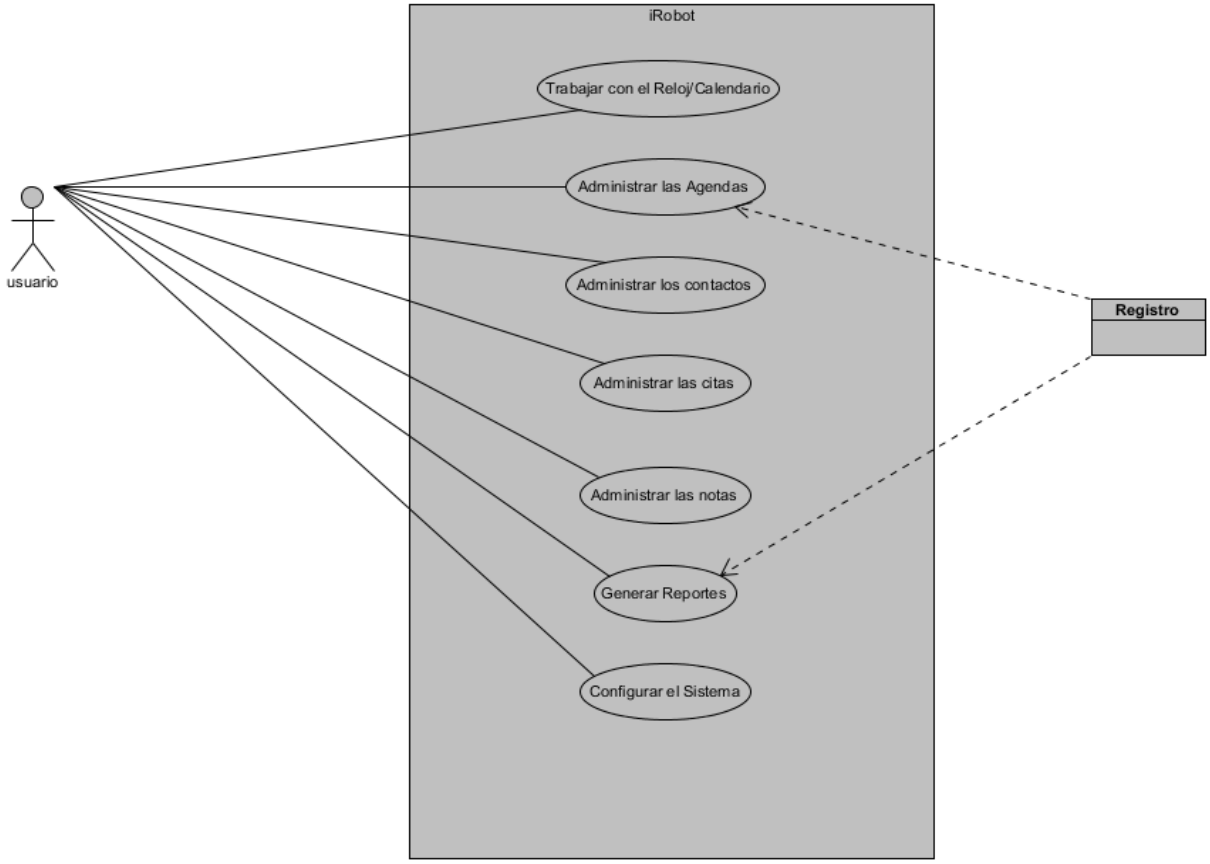


Figura 2. Este es el Diagrama General de Casos de Uso obtenido del análisis textual.

Escenarios AGENDAS

Nombre del caso de uso: Agregar tema	**ID única:**
Area: Agenda Laboral o Personal	
Actor(es): Usuario, Archivo Pendientes/Terminados	
Descripcion: Permitir al usuario agregar un tema o actividad a la lista de temas pendientes o terminados, con los datos del tema como Título, Clasificación, Fecha, Código Único y Tiempo utilizado (en horas) que puede agregarse automática o manualmente.	
Evento desencadenador: El usuario hace clic en el botón Agregar Tema (+) de la Agenda Laboral o Personal	
Tipo de desencadenador: ☐ **Externo**	☐ **Temporal**
Pasos realizados (Ruta principal)	**Informacion para los pasos**
El usuario da clic en botón Agregar de la Agenda	**actual, Código Único, Agenda Laboral o Personal, Temas Pendientes o Termi**
tana emergente y presionar Aceptar. La fecha y C	**Título, Clasificación, Notas, Horas Trabajadas**
la Etiqueta 'En edición', de lo contrario se pide a	**Activar Temporizador, Tiempo en horas para carga manual**
suario guarda la información al dar clic en 'Guard	**Archivo Pendiente o Terminado**
Pre-condiciones: El usuario se encuentra en las Agendas y no está activo el Temporizador	
Post-condiciones: El usuario agregó el tema y el temporizador puede estar activo	
Suposiciones: El usuario va a trabajar en un tema nuevo o capturar uno que olvidó capturar previamente	
Requerimientos cumplidos: El tema es agregado al archivo correspondiente y visualizado en la Agenda	
Cuestiones pendientes:	
Prioridad:	
Riesgo	

Figura 3. Este escenario corresponde al caso de uso “Agregar Tema”. Podemos identificar fácilmente los elementos de la interfaz y la manera en la que se comportarán al agregar un tema a la Agenda Laboral o Personal.

Nombre del caso de uso: Eliminar tema	**ID única:**
Area: Agenda Laboral o Personal	
Actor(es): Usuario, Archivo Pendientes/Terminados	
Descripcion: Permitir al usuario eliminar un tema o actividad de la lista de temas pendientes o terminados	
Evento desencadenador: El usuario hace clic en el botón Eliminar Tema (-) de la Agenda Laboral o Personal	
Tipo de desencadenador: ☒ **Externo**	☒**Temporal**
Pasos realizados (Ruta principal)	**Informacion para los pasos**
El usuario selecciona un tema de la Agenda	**Tema de la lista de Pendientes o Terminados**
El usuario da clic en botón 'Eliminar' de la Agenda	**Código Único, Agenda Laboral o Personal, Temas Pendientes o Terminados**
io debe confirmar la eliminación del tema con 'A	**Pregunta de confirmación de eliminación**
na es eliminado de la lista correspondiente. El Ar	**Lista de pendientes o terminados**
suario guarda la información al dar clic en 'Guard	**Archivo Pendiente o Terminado**
Pre-condiciones: El usuario se encuentra en las Agendas , no está activo el Temporizador y hay un tema seleccionado	
Post-condiciones: El usuario eliminó el tema de la lista y del archivo correspondientes.	
Suposiciones: El usuario va a eliminar un tema que ya no necesita tener almacenado	
Requerimientos cumplidos: El tema es eliminado del archivo correspondiente y ya no es visualizado en la Agenda	
Cuestiones pendientes:	
Prioridad:	
Riesgo	

Figura 4. Este escenario corresponde al caso de uso “Eliminar Tema”. Corresponde a la sección de la Agenda Laboral o Personal.

Nombre del caso de uso: Editar tema	**ID única:**
Area: Agenda Laboral o Personal	
Actor(es): Usuario, Archivo Pendientes/Terminados	
Descripcion: Permitir al usuario editar un tema o actividad de la lista de temas pendientes o terminados	
Evento desencadenador: El usuario hace clic en el botón Editar Tema (/) de la Agenda Laboral o Personal	
Tipo de desencadenador: 🗀 **Externo**	🗀**Temporal**
Pasos realizados (Ruta principal)	**Informacion para los pasos**
El usuario selecciona un tema de la Agenda	**Tema de la lista de pendientes o terminados**
El usuario da clic en botón 'Editar' de la Agenda	**Código Único, Agenda Laboral o Personal, Temas Pendientes o Terminados**
usuario modifica los datos en la ventana emerger	**Fecha, Título, Clasificación, Notas, Horas Trabajadas, Status**
n 'Aceptar' y los datos en la lista correspondient	**Lista de pendientes o terminados**
suario guarda la información al dar clic en 'Guarc	**Archivo Pendiente o Terminado**
Pre-condiciones: El usuario se encuentra en las Agendas , no está activo el Temporizador y hay un tema seleccionado	
Post-condiciones: El usuario actualizó el tema de la lista y del archivo correspondientes.	
Suposiciones: El usuario va a editar información de un tema que ya no es correcta.	
Requerimientos cumplidos: El tema es actualizado en el archivo correspondiente y visualizado en la Agenda	
Cuestiones pendientes:	
Prioridad:	
Riesgo	

Figura 5. Este escenario corresponde al caso de uso “Editar Tema”. Corresponde a la sección de la Agenda Laboral o Personal.

Nombre del caso de uso: Activar tema	ID única:
Area: Agenda Laboral o Personal	
Actor(es): Usuario	
Descripcion: Permitir al usuario activar un tema o actividad en la lista de temas pendientes o terminados para continuar acumulando tiempo.	
Evento desencadenador: El usuario hace clic en el botón Continuar Tema (Play) de la Agenda Laboral o Personal	
Tipo de desencadenador: Externo	Temporal
Pasos realizados (Ruta principal)	Informacion para los pasos
tema que desea continuar de la lista de temas pe	Agenda Laboral o Personal, Temas Pendientes o Terminados
suario da clic en botón Continuar Tema de la Age	Agenda Laboral o Personal, Temas Pendientes o Terminados
Pre-condiciones: El usuario se encuentra en las Agendas y no está activo el Temporizador y hay por lo menos un tema en la ag	
Post-condiciones: El temporizador está activo	
Suposiciones: El usuario cronometrará una actividad pendiente o activará nuevamente una terminada	
Requerimientos cumplidos: La actividad es cronometrada o activada nuevamente según sea el caso.	
Cuestiones pendientes:	
Prioridad:	
Riesgo	

Figura 6. Este escenario corresponde al caso de uso “Activar Tema”. Corresponde a la sección de la Agenda Laboral o Personal. Permite iniciar el temporizador para medir el tiempo empleado en una actividad nueva o pendiente.

Nombre del caso de uso: Pausar tema	**ID única:**
Area: Agenda Laboral o Personal	
Actor(es): Usuario	
Descripcion: **Permitir al usuario pausar temporalmente un tema o actividad en la lista de temas pendientes para realizar otra actividad diferente. El tiempo medido es agregado automáticamente al campo Tiempo del tema.**	
Evento desencadenador: El usuario hace clic en el botón Pausar Tema (Pause) de la Agenda Laboral o Personal	
Tipo de desencadenador: **Externo** **Temporal**	
Pasos realizados (Ruta principal)	**Informacion para los pasos**
usuario da clic en botón Pausar Tema de la Agen	**Agenda Laboral o Personal, Temas Pendientes o Terminados, Temporizador**
Pre-condiciones: El Temporizador está activo y un tema está seleccionado.	
Post-condiciones: El temporizador se desactiva y el tiempo medido es agregado al campo Tiempo del tema. El Estado del tema	
Suposiciones: El usuario realizará otra actividad.	
Requerimientos cumplidos: El tiempo medido es agregado al tema.	
Cuestiones pendientes: Activar el temporizador de inactividad.	
Prioridad:	
Riesgo	

Figura 7. Este escenario corresponde al caso de uso “Pausar Tema”. Corresponde a la sección de la Agenda Laboral o Personal y hace referencia a la opción que tiene el usuario de pausar temporalmente el temporizador para realizar alguna otra actividad, mientras se suma el tiempo acumulado con el empleado hasta antes de la pausa.

Nombre del caso de uso: Detener tema	**ID única:**
Area: Agenda Laboral o Personal	
Actor(es): Usuario, Archivo de Terminados	
Descripcion: **Permitir al usuario terminar un tema y agregarlo a la lista de temas terminados, agregando automáticamente el tiempo medido si el temporizador está activo. El usuario puede posteriormente guardar la información en el archivo de Temas Terminados.**	
Evento desencadenador: El usuario hace clic en el botón Detener Tema (Stop) de la Agenda Laboral o Personal	
Tipo de desencadenador: ☒ **Externo**	☒**Temporal**
Pasos realizados (Ruta principal)	**Informacion para los pasos**
ma de la lista de pendientes en caso de que el Te	**Agenda Laboral o Personal, Temas Pendientes o Terminados**
ısuario da clic en botón Detener Tema de la Ager	**Agenda Laboral o Personal, Temas Pendientes o Terminados, Temporizador**
lista de temas terminados. Si el temporizador es	**Agenda Laboral o Personal, Temas Pendientes o Terminados, Temporizador**
macenar esta información deberá hacer clic en G	**Agenda Laboral o Personal, Temas Pendientes o Terminados**
Pre-condiciones: El Temporizador puede estar activo o inactivo y un tema es seleccionado.	
Post-condiciones: El temporizador se desactiva y el tiempo medido es agregado al campo Tiempo del tema. El tema es movido	
Suposiciones: El usuario terminó con esta actividad.	
Requerimientos cumplidos: El tiempo medido es agregado al tema y el tema tiene el Estado Terminado	
Cuestiones pendientes:	
Prioridad:	
Riesgo	

Figura 8. Este escenario corresponde al caso de uso “Detener Tema”. Corresponde a la sección de la Agenda Laboral o Personal. Hace referencia al momento en el que una actividad es terminada y se requiere que sea movida al listado de Temas Terminados con la adición del tiempo acumulado en el campo correspondiente.

Diagramas de Secuencia

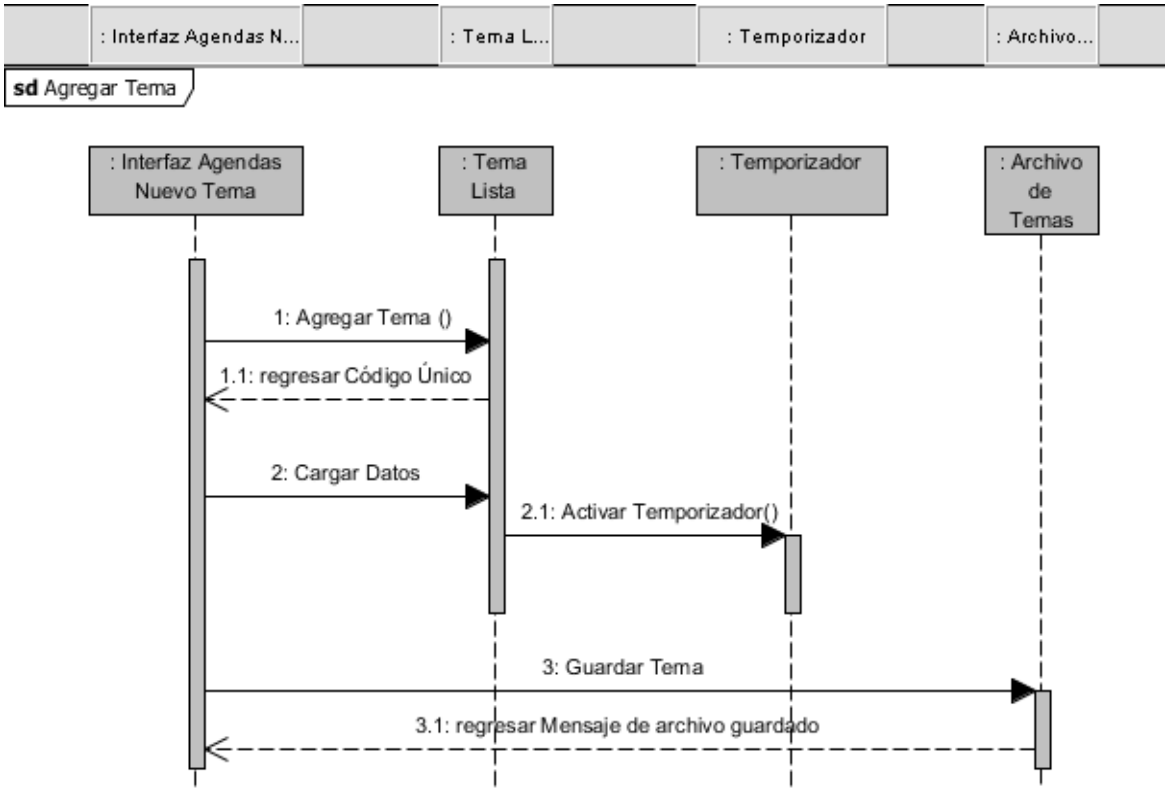


Figura 9. Diagrama de secuencia para Agregar Tema

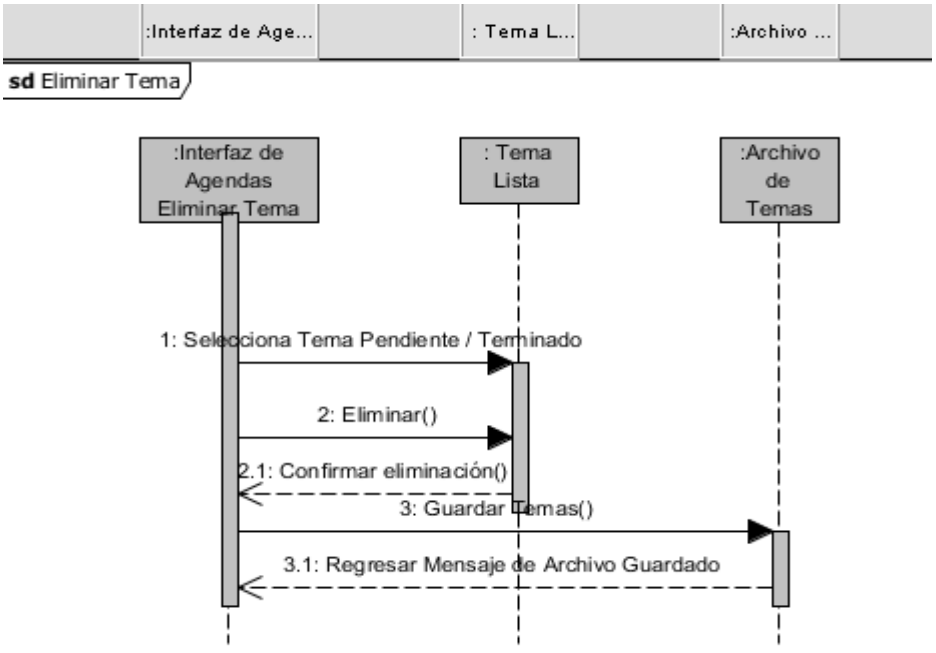


Figura 10. Diagrama de secuencia para Eliminar Tema

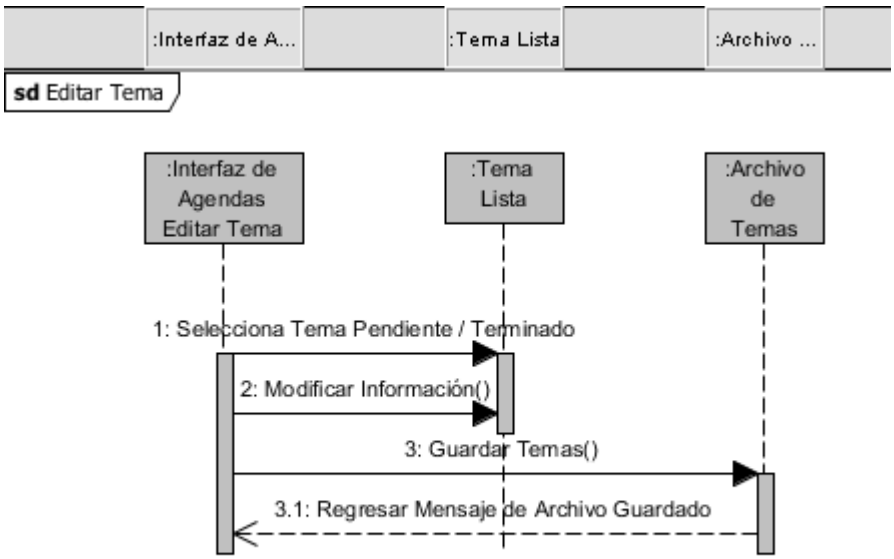


Figura 11. Diagrama de secuencia para Editar Tema

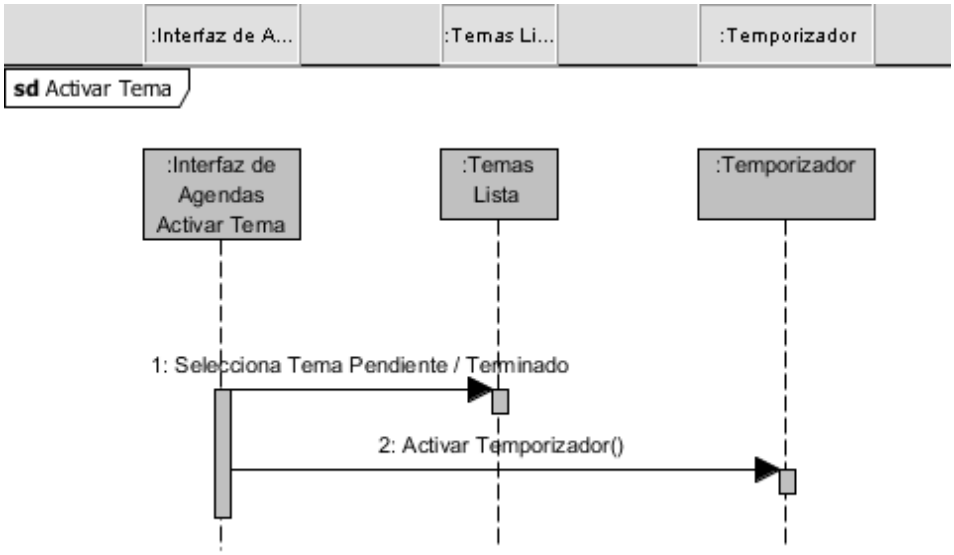


Figura 12. Diagrama de secuencia para Activar Tema

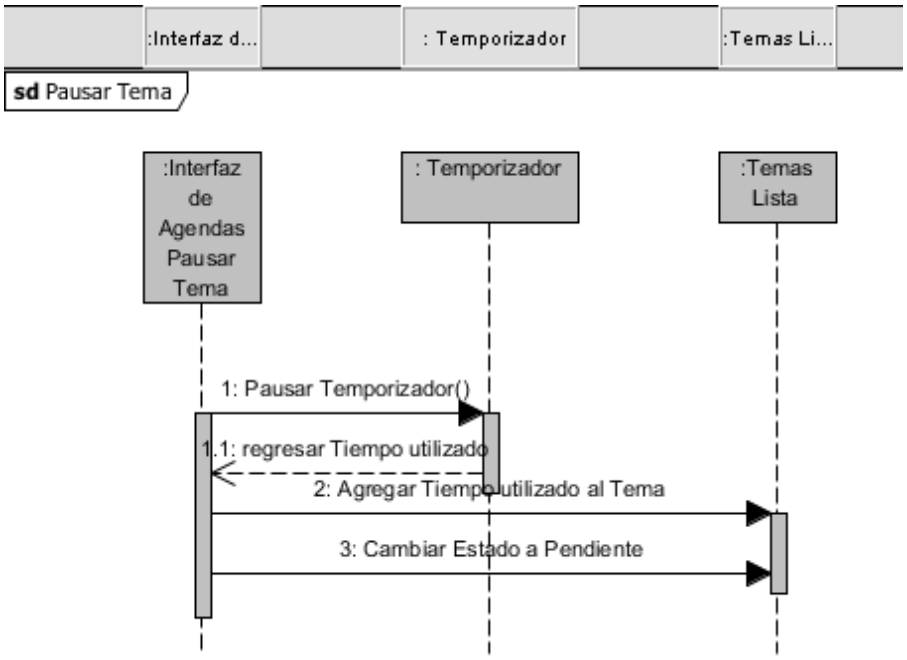


Figura 13. Diagrama de secuencia para Pausar Tema

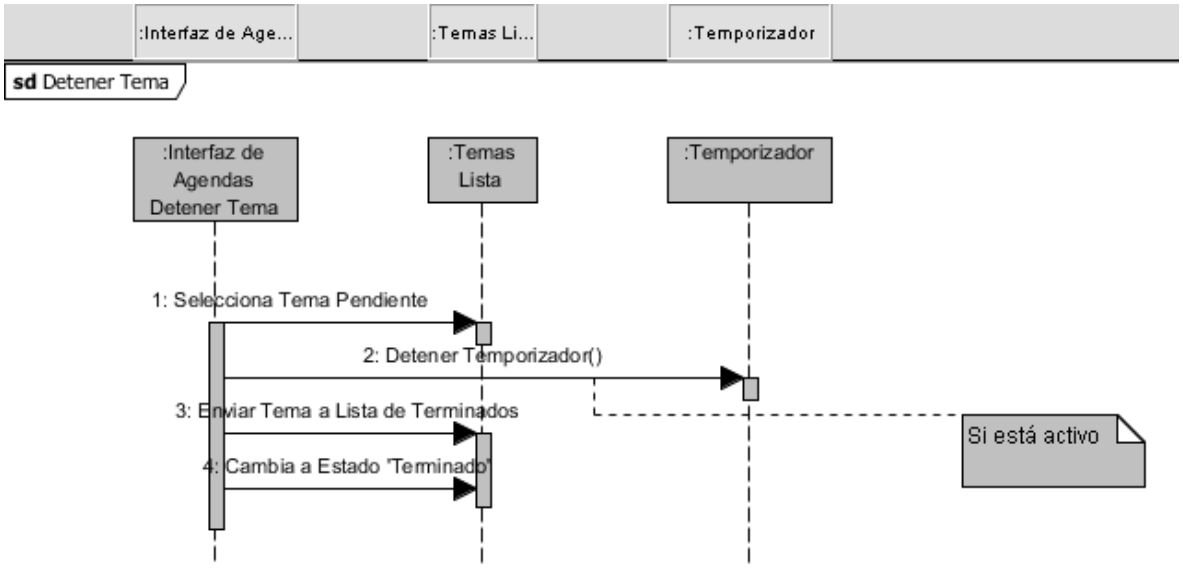


Figura 14. Diagrama de secuencia para Detener Tema

Diagrama de Clases

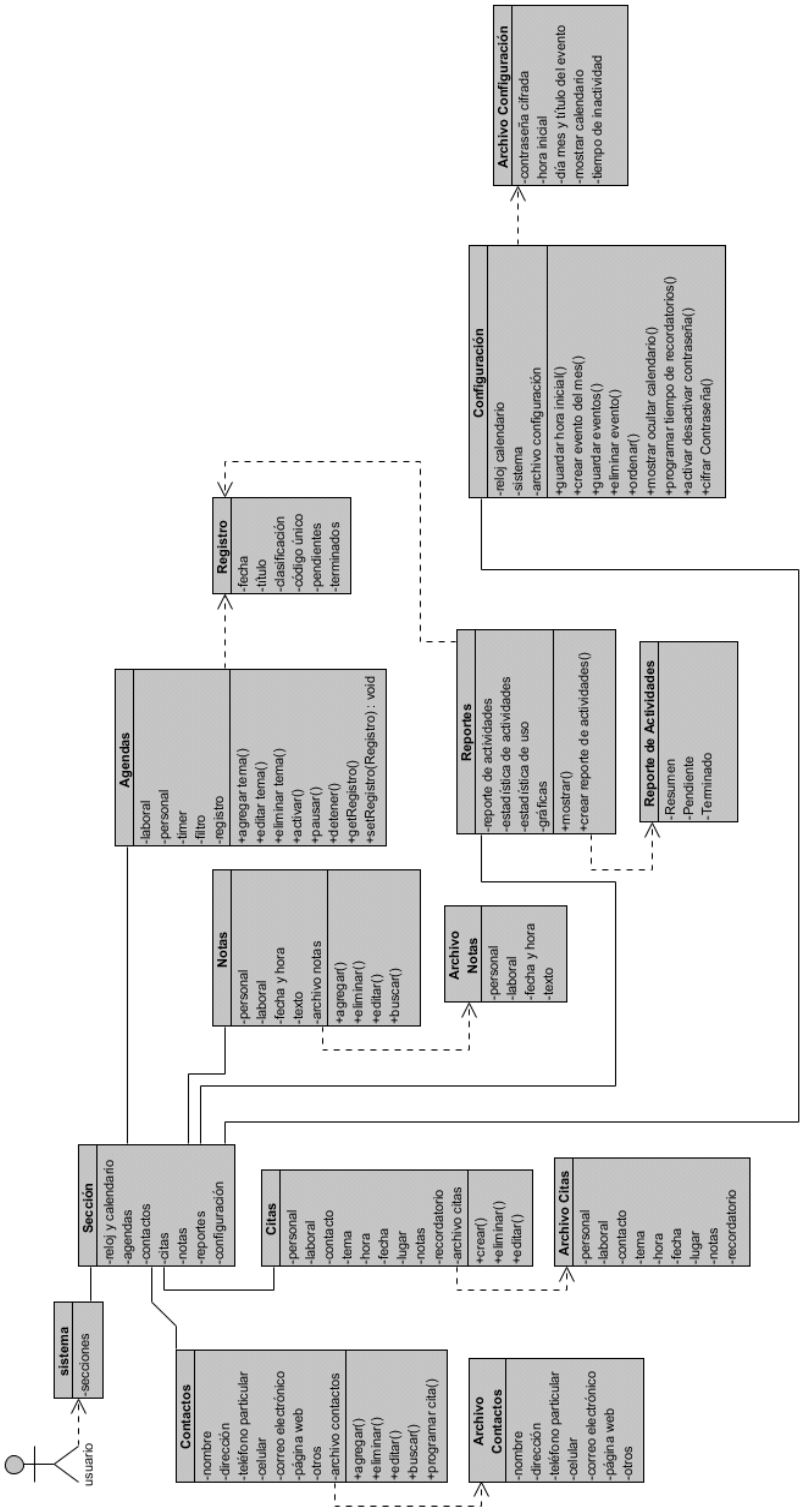


Figura 15. Diagrama de clases, con atributos y relaciones.

Implementación

Interfaces

La interfaz del sistema es una interfaz simple, visualmente atractiva y fácil de usar. Cuenta con siete secciones que proporcionan 8 herramientas para una mejor administración del tiempo. Las secciones del sistema son: Reloj/Calendario, Agenda, Contactos, Citas, Notas, Reportes y Configuración.
Las diferentes secciones o interfaces pueden ser accedidas mediante una barra de iconos colocada en el extremo derecho de la ventana.
Al seleccionar uno de los iconos, los demás se “oscurecen”.

Esta barra está disponible en todo momento, excepto cuando el botón de Ocultar/Mostrar marquesina se ha presionado para mostrar únicamente el reloj, que es la interfaz mínima que se puede tener.
A continuación se describirán brevemente cada una de las secciones / interfaces, y posteriormente se describirá su funcionamiento.

Interfaz Reloj/Calendario

Esta es la interfaz principal y proporciona al usuario la información que se obtiene de un reloj de escritorio: la hora actual (con segundero), el día de la semana, la fecha y el número de semana.
También posee un calendario que sirve no solo para visualizar los días del mes o la fase lunar, sino para administrar los eventos del año a manera de agenda, mientras una marquesina despliega constantemente información sobre la interfaz actual.
La marquesina es un elemento visual práctico y utilizado en todas las vistas del programa. Se emplea principalmente para desplegar información sobre dicha vista, en el caso del Reloj/Calendario muestra los eventos del mes, para las demás vistas muestra datos del elemento seleccionado en las listas correspondientes. En Configuración muestra información del autor al hacer clic en “Acerca de”.

La interfaz tiene dos botones, uno de “apagado” y el otro que en un primer paso despliega la marquesina y los accesos directos, y en el segundo un calendario propiamente dicho.

Esta interfaz es reubicable, al arrastrarla desde cualquier número indicador de la hora. Para hacerlo más evidente el cursor cambia al colocarlo sobre éstos.
Al hacer doble clic sobre el día, el calendario muestra el mes actual (en caso de que se visualizara otro).
Para la sección del calendario, existen diversas funciones.
En la parte superior del mismo se muestra el mes y año actualmente en consulta y puede ser cambiado al hacer clic en cualquiera de las “flechas” izquierda o derecha para retroceder o avanzar mes a mes respectivamente.
Los días se muestran en 6 colores distintos:

Día del mes anterior/siguiente

Día ya pasado del mes

Día actual del mes

Día del mes aún por venir

Día con evento

Día actual del mes con evento

Cuando el mes corresponde al de la semana santa del año mostrado, en la marquesina se desplegarán el día que corresponde al jueves y viernes santos así como en el calendario aparecerán dichos días como días con evento.

Adicionalmente se muestra una imagen de la fase lunar actual y un porcentaje que corresponde a la edad mensual de la misma, donde un 100% corresponde a la luna nueva y 50% a la luna llena.

Al hacer doble clic en cualquier día del mes, se muestra automáticamente la ventana de “Agregar Evento” donde se captura el nombre del evento y al presionar aceptar se muestra la ventana de configuración de eventos.

La ventana de Agregar Evento tiene los siguientes campos:

Mes: Permite seleccionar el mes para el evento. Por default se muestra el mes actual o en el caso de doble clic en el calendario, el mes visualizado correspondiente.
Día: Permite seleccionar el día del evento. Por default se muestra el día actual o en el caso de doble clic en el calendario, el día en que se hizo doble clic.
Evento: Es el campo donde se captura el nombre del evento.
Repetir Anualmente: Si la casilla está seleccionada, entonces el evento será agregado como uno que debe repetirse cada año en la misma fecha. Estos eventos se almacenan en un archivo adicional y se identifican con un ‘*’
Cancelar: Cancela esta acción y cierra la ventana de “Agregar Evento”
Agregar: Muestra la ventana “Configuración de Eventos” que se describe a continuación.

Ventana de Configuración de Eventos.

Esta ventana puede accederse al hacer clic en el botón “Agregar” de la ventana “Agregar Evento” o del botón “Config. Eventos” de la interfaz de Configuración (véase más adelante)
Esta ventana muestra un listado de los eventos que tiene almacenados para el año en curso, incluyendo aquellos que se repiten anualmente y que están identificados con un ‘*’.
La interfaz cuenta con cuatro botones:

 Agregar: Muestra la ventana “Agregar Evento” que permite agregar un evento al listado de eventos. El evento se coloca automáticamente en el día y mes que le corresponde en el listado en un orden ascendente.

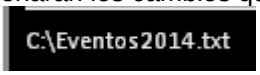
 Eliminar: Para eliminar un evento, debe seleccionarse primero en la lista y posteriormente presionar este botón. Después de una confirmación, el evento será eliminado de la lista únicamente y permanentemente al presionar el botón Guardar.

 Editar: Al seleccionar un evento del listado y después este botón, se mostrará una ventana idéntica a la de Agregar Evento, con la diferencia de que ahora está editando un evento existente en lugar de agregar uno nuevo. Puede cambiar el mes, día, texto y si se trata o no de un evento repetitivo anualmente.

 Guardar: Este botón almacena en el archivo de eventos todo lo que se encuentre actualmente en el listado de eventos.

Cerrar: Este botón cierra esta ventana. Si ha efectuado cambios en el listado y no los ha guardado, el programa le preguntará si desea guardarlos antes de cerrarse.

En la parte inferior de la ventana se muestra el nombre del archivo de eventos que se encuentra cargado actualmente y donde se almacenarán los cambios que efectúe en el listado.



Una vez que ha seleccionado la fecha (si es necesario) y escrito el nombre del evento en la ventana emergente, deberá dar clic en Guardar para que los cambios se reflejen y la información se almacene en el archivo correspondiente.
Al hacerlo, el día cambia su configuración a “Día con evento” como se describió previamente, mostrándose esta información en la marquesina y almacenándola en el archivo de Eventos.
Si un evento fue creado con la opción de “Repetir Anualmente” y el mes mostrado corresponde al de algún evento anual, éste es mostrado en la marquesina y marcado como “Día con evento” en el calendario. La marca para identificar un evento anual es un “*” al final del nombre del evento.

Interfaz Agendas

Para llevar una mejor administración del tiempo, es necesario llevar una buena administración de las tareas que se realizan diariamente. Las agendas facilitan dicha administración mediante un control muy exacto del tiempo empleado en cada tarea o actividad.
Las agendas, son listas de tareas que pueden clasificarse en 4 diferentes grupos: Agenda Laboral Pendientes, Agenda Laboral Terminados, Agenda Personal Pendientes y Agenda Personal Terminados. Como su nombre lo indica, podemos guardar tareas o temas laborales y personales y a su vez clasificarlos como pendientes cuando son creados por primera vez o trabajados pero sin terminarlos. Cuando una tarea se termina, es movida del listado de pendientes a su correspondiente de terminados. Y a su vez, puede ser devuelta a pendientes cuando se tiene que re-trabajar.
La interfaz cuenta con dos secciones principales, en la sección inferior se muestra uno de los 4 listados de tareas mencionados, dos botones para seleccionar Pendientes o Terminados en la orilla inferior. Y del costado derecho se encuentran 6 “botones” que son comunes en el resto de interfaces excepto por los reportes y la configuración. Estos botones son:

 -Personal

 -Laboral

 -Agregar

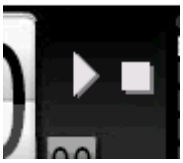
 -Eliminar

 -Editar

 -Guardar

Los primeros permiten seleccionar si se muestran las agendas (y contactos, citas, notas) personales o laborales.
El botón agregar muestra una ventana que permite cargar un nuevo tema con todos los datos que se muestran en la lista de tareas.
El botón eliminar permite quitar del listado el tema que está seleccionado actualmente. Se solicitará una confirmación del usuario antes de eliminar cualquier información.
El botón editar muestra la ventana con los datos del tema para modificarlos y guardar los cambios.
El botón guardar permite almacenar en un archivo el listado de tareas tal cual se muestra en la interfaz.
Cuando se selecciona algún tema o tarea del listado, los datos principales del mismo son mostrados en la marquesina.

En la parte superior de esta interfaz los cuadros que normalmente muestran la hora actual, se transforman en las horas, minutos y segundos de un cronómetro que puede ser utilizado tal cual o para medir el tiempo que se emplea en uno o más temas que se seleccionen del listado de tareas.
Esta opción existe con la finalidad de medir el tiempo empleado por el usuario en diversas tareas que realiza y registra con el fin de optimizar y administrar de una mejor manera su tiempo.
Además de los dígitos, se muestran dos botones similares al de los reproductores de audio y video actuales: Play y Stop.



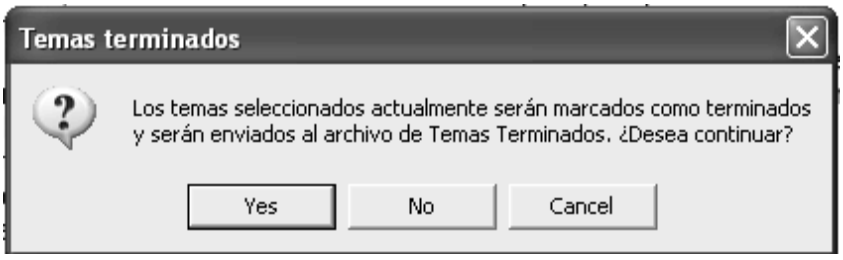
Como se intuye fácilmente al seleccionar un tema del listado y presionar Play, el cronómetro inicia y el botón cambia a Pausa. Mientras esté activo este botón el tiempo se estará midiendo continuamente hasta que sea presionado nuevamente (como Pausa) o se presione el botón Stop.

Ambos botones agregarán el tiempo, en minutos, que ha sido medido en el campo “Tiempo” de la tarea seleccionada.

La diferencia entre estos botones es que el botón Pausa, seguirá dejando el tema en la lista de Pendientes, con el tiempo adicionado.
El botón Stop detendrá completamente el tema, considerando que ha sido terminado, quitándolo del listado de pendientes y colocándolo en el listado de Terminados. Obviamente con el tiempo adicionado.
Cuando se visualiza el listado de temas Terminados (presionando el botón correspondiente), al presionar el botón Play el tema se mueve a la lista de Pendientes, previa confirmación del usuario.
Y si se presiona el botón Stop, no se hace nada, pues el tema ya se encuentra como Terminado. Esto es informado al usuario.
Si el cronómetro se encuentra corriendo, es posible cambiar de interfaz sin que esto le afecte.
Cuando un tema se detiene o termina con el botón Stop, el cronómetro es reiniciado a cero. Si se presiona el botón Pausa, el tiempo medido sigue apareciendo en el cronómetro, pero es reiniciado a ceros cuando se presiona nuevamente Play.
Es posible seleccionar más de un tema de los listados y activar el cronómetro. Cuando se presiona Pausa, el tiempo medido es dividido entre el número de temas seleccionados. Esto por la

imposibilidad real de ejecutar varias tareas exactamente al mismo tiempo y con el fin de una distribución más “justa y real” del tiempo empleado en las tareas.

Si hay más de un tema seleccionado al presionar “Stop”, los temas seleccionados serán enviados al listado de temas terminados y éste será mostrado.

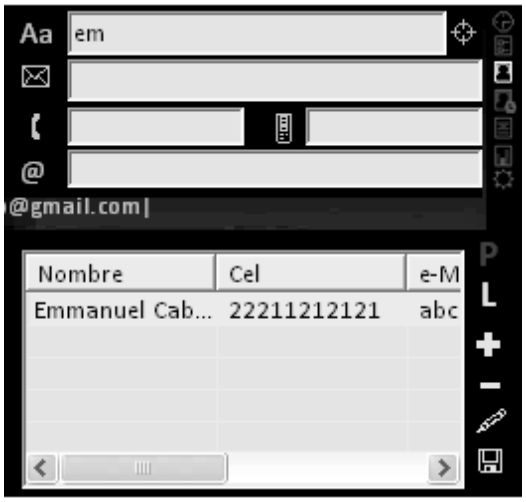


Interfaz Contactos

Para poder tener a la mano la información de contacto de sus conocidos tanto del trabajo como de su vida diaria, el programa tiene una interfaz de contactos.

Esta interfaz cuenta también con dos secciones. En la inferior podemos visualizar un listado de los contactos ya sea Personales o Laborales que se encuentran actualmente almacenados en el archivo de contactos correspondiente.
En esta sección tenemos los mismos seis botones de la Agenda, realizando las mismas acciones pero ahora con los datos de contactos.
En la parte superior se encuentran los datos del contacto: Nombre, Dirección, Teléfono, Celular y Correo electrónico.
Para agregar un tema, se escriben los datos en los campos y se presiona el botón Agregar. Para eliminar un contacto, se selecciona de la lista y se presiona el botón Eliminar.
Para editarlo, es necesario seleccionar el contacto y presionar Editar, al hacerlo, los datos se colocan en la parte superior, el usuario los modifica y posteriormente debe presionar el botón Guardar para que los cambios sean almacenados.
Cuando un contacto se selecciona, sus datos son mostrados en la marquesina.

En la parte superior se encuentra el botón 
Este sirve para buscar un contacto en específico mediante cualquier campo.
Para usarlo, debe darse clic y posteriormente escribir los datos de búsqueda en uno o varios de los campos de la parte superior. Al ir escribiendo automáticamente se irán filtrando los datos que coincidan en el listado inferior. Para desactivar la función solo debe darse clic en el botón nuevamente.



El botón Guardar almacena la información del contacto en su archivo correspondiente, Laboral o Personal.

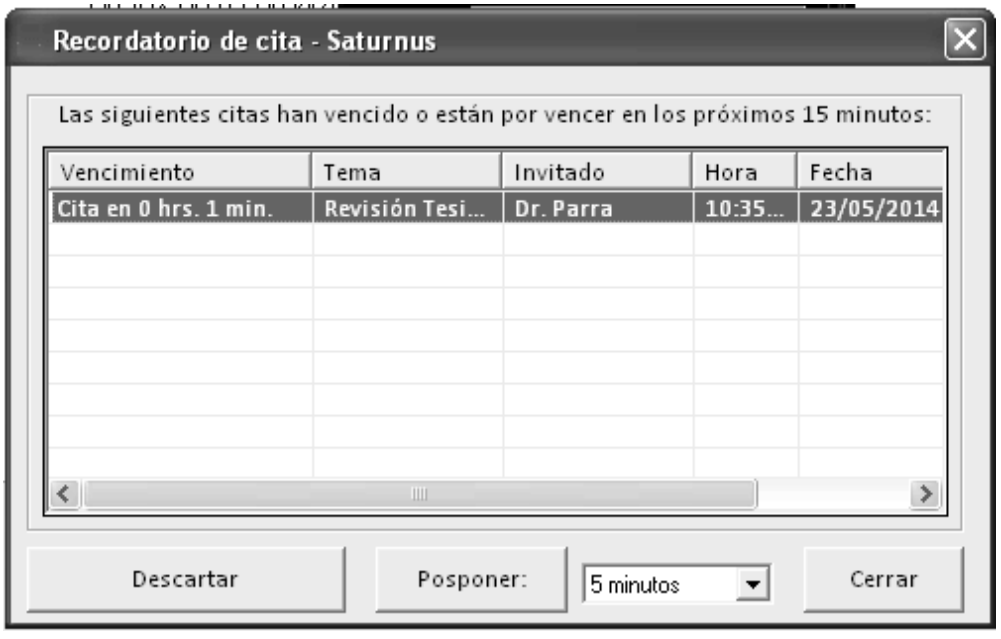
Interfaz Citas

Una administración de su tiempo efectiva no estaría completa sin una interfaz de citas, donde podrá anotar las reuniones a las que debe asistir o trámites que deba hacer. La fortaleza de esta interfaz no es tanto la cita en sí sino los recordatorios que le permite agregar.
Esta interfaz le permite la creación de citas muy sencillas, las cuales contienen como información el Tema, el Invitado, el Lugar, la Fecha y la Hora.

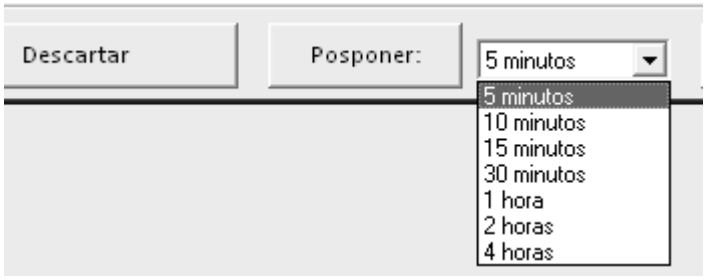
Estos datos se cargan en la parte superior de la interfaz y se agregan con el botón .
En la parte inferior de la interfaz se muestra un listado con las citas almacenadas y nuevamente como en las interfaces anteriores, están disponibles los botones Agregar, Eliminar, Editar y Guardar, así como los de Personal y Laboral.
Las citas se clasifican nuevamente en laborales y personales, almacenándose en su archivo correspondiente al dar clic en Guardar.

Para eliminar una cita, debe seleccionarse primero y posteriormente presionar el botón . Antes de eliminarse la cita, se le solicitará una confirmación.

Para editarla, debe seleccionarse y al presionar los datos son colocados en los campos correspondientes de la parte superior y están activos para editarse. Una vez modificada la información debe darse clic en para que los cambios surtan efecto. Si no desea guardar los cambios, simplemente presione nuevamente el botón .
Cuando una cita está por vencer, se muestra un recordatorio de la cita, en el tiempo que se haya configurado en la interfaz de Configuración (v. Interfaz de Configuración).



La ventana que se muestra permite programar el tiempo en el que se mostrará nuevamente el recordatorio, 5, 10, 15, 30, 60, 120 y 240 minutos son las opciones a elegir. Así como también puede elegirse la opción “descartar” que considerará la cita como tomada y no se mostrará nuevamente el recordatorio.

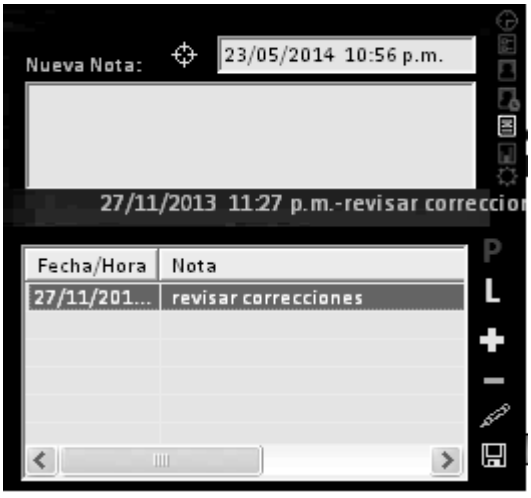


En el listado de citas, aquellas que hayan sido descartadas serán marcadas con un “*” en el Tema.
Si no se elige ni la opción Posponer ni Descartar, y se da clic en Cerrar, entonces el recordatorio estará apareciendo cada minuto hasta que se elija alguna de las dos opciones mencionadas.
Es posible también buscar citas por cualquiera de los campos de las mismas.

Se debe dar clic en el botón  y entonces comenzar a escribir en cualquier campo y entonces se irán mostrando en la lista las citas que coincidan con los datos ingresados.
Para regresar al modo normal, se debe dar clic en el botón nuevamente.

Interfaz Notas

Finalmente, para respaldar la información que desea recordar pero puede olvidar, la interfaz de notas le proporciona una herramienta muy rápida y sencilla en la cual podrá hacer anotaciones de cualquier tipo, páginas web, frases, cosas por investigar, libros por leer, etc.



Esta es quizá la interfaz más sencilla. Cuenta con los botones básicos de Agregar, Eliminar, Editar y Guardar, y también con los de Personal y Laboral.
Los campos que pueden capturarse para las notas son Fecha/Hora y la nota en sí. Estos campos se encuentran en la parte superior de la interfaz y el listado de notas almacenadas en la parte inferior de la misma.

Para agregar una nota simplemente debe escribirse en el campo correspondiente “Nueva Nota” y presionar el botón , el campo de fecha y hora contienen la información actual, aunque si se desea puede modificarse directamente en el campo esta información.
Igualmente las notas son guardadas en archivos separados para las notas laborales y las notas personales.

Para eliminar una nota, se debe seleccionar del listado y presionar el botón . Al hacerlo se le solicitará una confirmación.

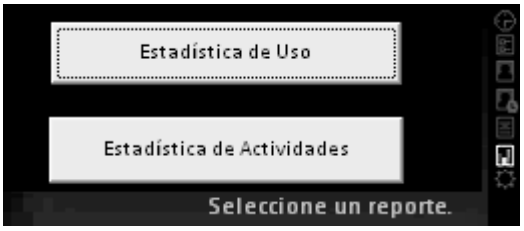
Para editar una nota, selecciónela del listado y de clic en esto colocará el texto de la nota en su campo correspondiente en la parte superior y podrá modificar el texto a su antojo para posteriormente almacenar los cambios con el botón . Si no desea guardar los cambios, simplemente presione el botón nuevamente y la nota permanecerá intacta.

Puede también buscar en sus notas cualquier texto que desee. Presione el botón para activar la función y al igual que en las demás interfaces previas, escriba directamente el texto a buscar en el campo de texto de la nota. Al hacerlo, todas las notas que contengan ese texto serán filtradas y mostradas automáticamente en el listado.

Para desactivar la función presione nuevamente el botón .
El listado mostrará las notas en el orden en el que fueron creadas.

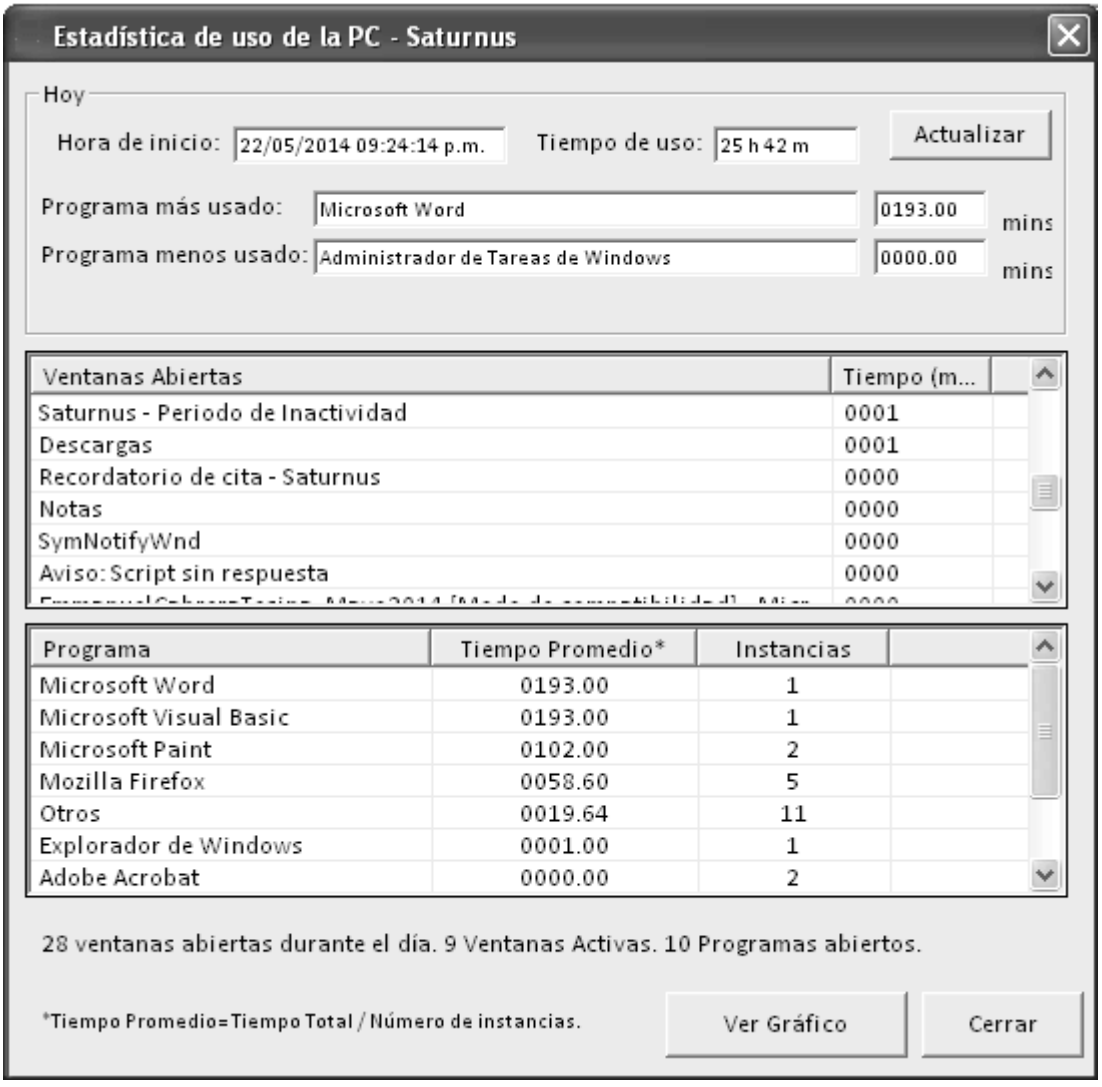
Interfaz de Reportes

Si necesita saber en qué y cómo ha estado ocupando el tiempo que ha estado frente a su computadora, la interfaz de reportes le proporciona la información resumida para analizarla y en su caso reconsiderar los tiempos que emplea en cada actividad.



Esta interfaz cuenta con dos botones que abren dos diferentes ventanas con información específica sobre el Uso y las Actividades registradas por el programa.
Estos botones son “Estadística de Uso” y “Estadística de Actividades”

Estadística de Uso



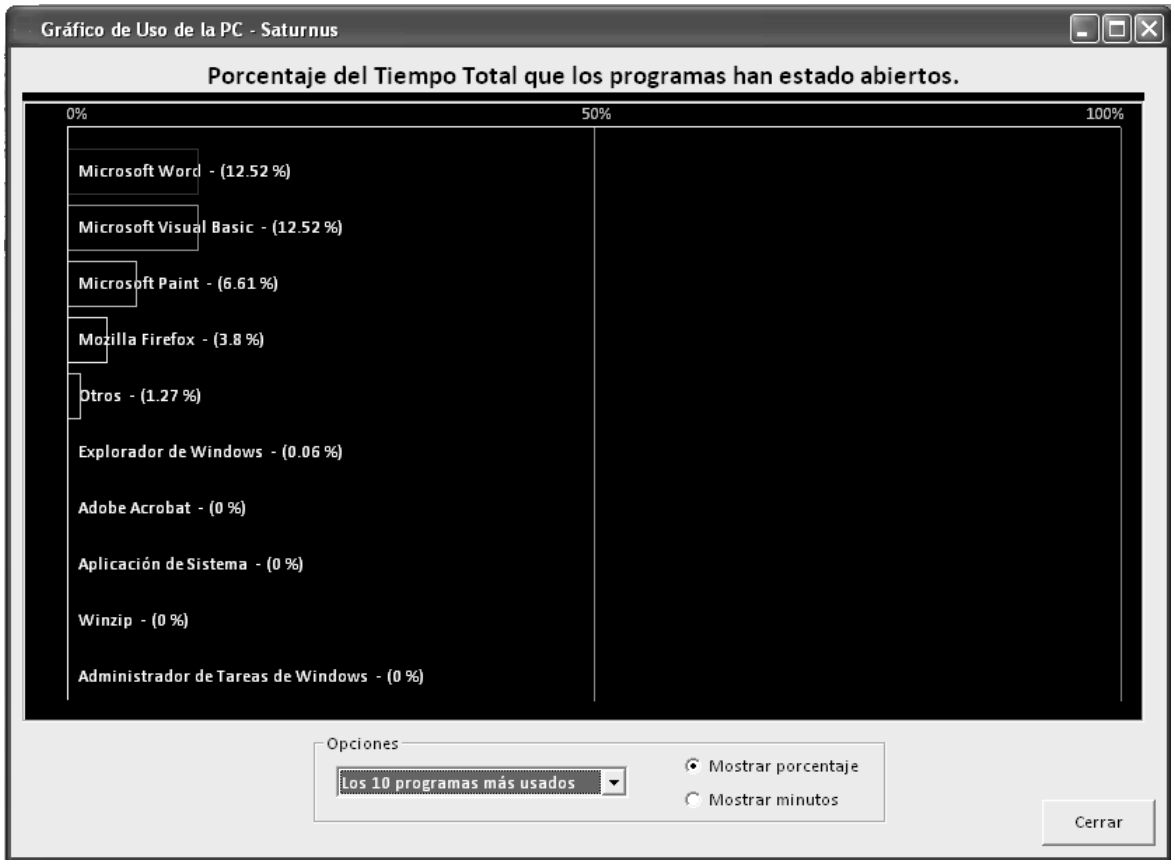
La estadística de uso se encarga de recolectar información sobre los programas y las ventanas que abre durante el día, la hora de inicio de sesión, el tiempo total de uso, los programas más y menos utilizados. Esta información también se puede mostrar de manera gráfica para facilitar la lectura e interpretación de la misma.
La interfaz se divide en dos secciones principales, la primera, la superior, le muestra la fecha y hora del inicio de uso de la PC y el tiempo total de uso.
También aquí se muestra el programa más usado y el menos usado en cuanto a tiempo. Y para tener la información a último estado, hay un botón de Actualizar que refrescará la información mostrada.
En la sección inferior de la interfaz, se muestran dos listados, el primero contiene el título de todas las ventanas que se han abierto durante el día, así como el tiempo en minutos que estuvieron disponibles o cargadas en memoria (no necesariamente activas). Esto es útil para conocer las ventanas en las que normalmente trabaja y quizá administrar de una manera distinta su tiempo y recursos. Este listado está ordenado de mayor a menor respecto al tiempo de permanencia de las ventanas.

Inmediatamente debajo de este listado, se muestra otro con información de los programas que ha utilizado durante el día. La diferencia con el listado anterior es que en este, las ventanas son

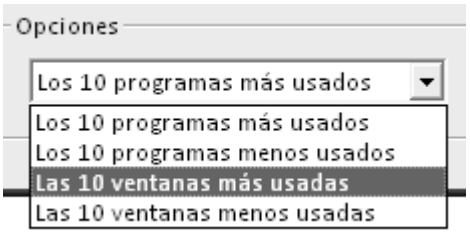
agrupadas respecto al programa al que pertenecen, para tener un cálculo más certero sobre el uso de su PC. Este cálculo se hace dividiendo el tiempo total de las ventanas abiertas por programa entre el número de ventanas, lo cual nos da un tiempo de uso del programa en sí.
De esta manera, podemos saber cuál es el programa más usado durante el día, y cuando se encuentra en su trabajo, estos resultados pueden sorprender al usuario.
En la parte inferior de estos listados, se le informa del total de programas y ventanas abiertos durante el día, así como el de ventanas activas.

En la parte inferior, al lado del botón cerrar, existe un botón 

que lo llevará directamente a la ventana que muestra un gráfico de barras de la información de esta interfaz.

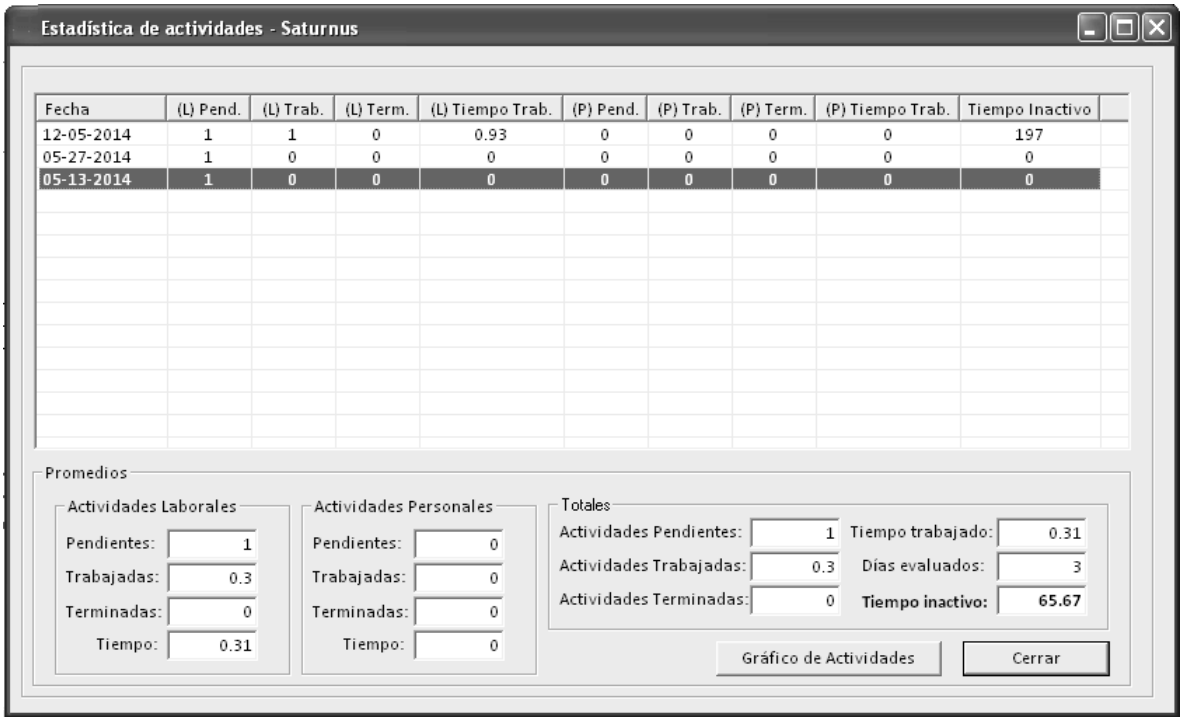


La ventana del gráfico solamente tiene un área de opciones que le permite seleccionar si se muestran minutos o porcentaje y qué se gráfica: los 10 programas más usados, los 10 programas menos usados, las 10 ventanas más usadas o las 10 ventanas menos usadas.



El botón cerrar cierra únicamente esta ventana.

Estadística de Actividades



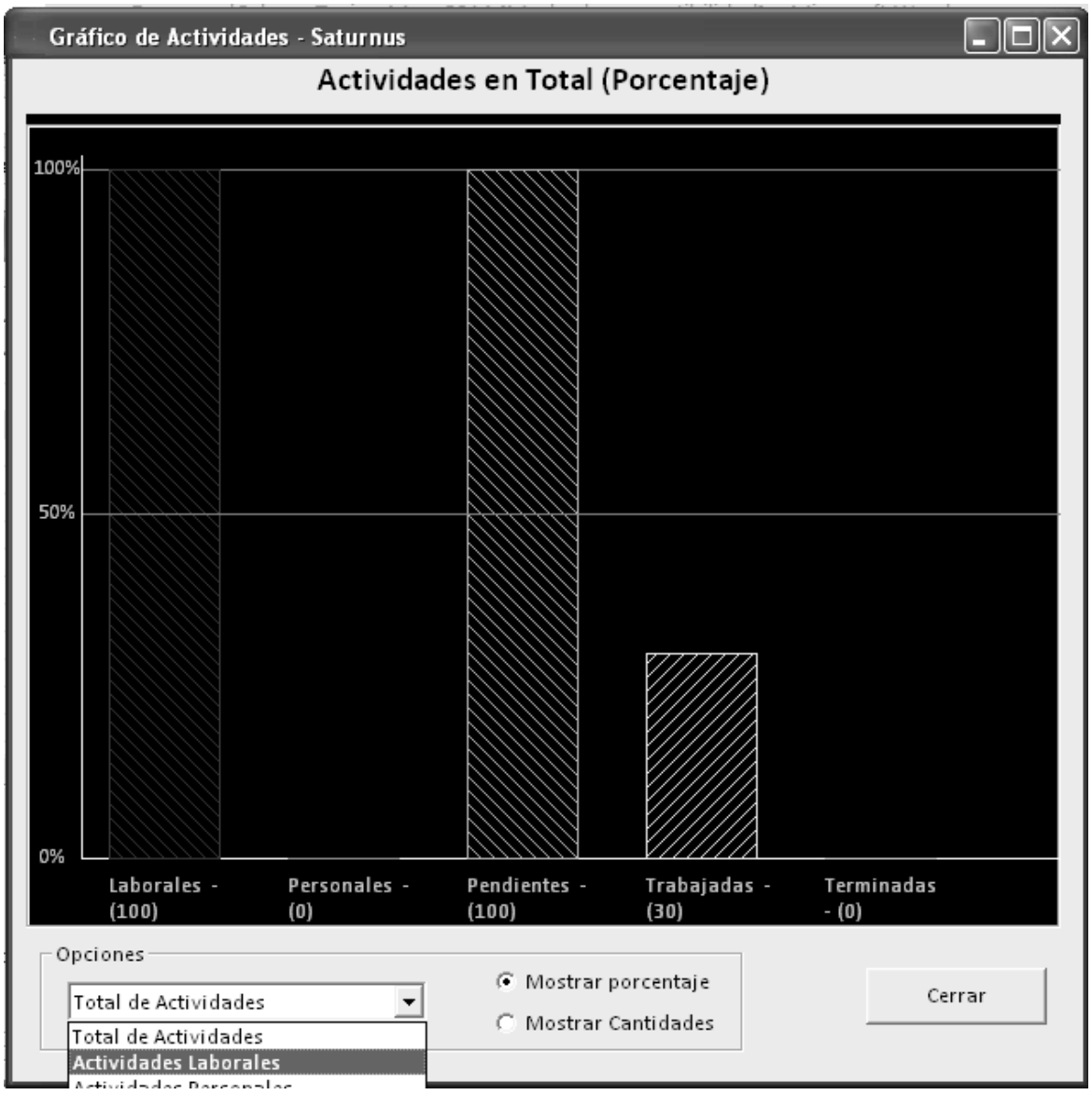
La estadística de actividades le mostrará información referente a las tareas o actividades de su agenda, tanto laboral como personal. De esta manera podrá conocer cómo se han desarrollado sus actividades los últimos días o meses, saber cuántas tareas terminó, cuántas tuvo nuevas, y algo muy importante, cuánto tiempo ha permanecido inactivo durante esos días.

El tiempo de inactividad se calcula sumando los minutos en los que ninguna tarea está cronometrándose, o en pocas palabras, mientras el botón Play y no el de Pausa esté visible.

Esto es útil para evaluar cuánto tiempo puede ser potencialmente empleado en alguna actividad que ha dejado de hacer “por no tener el tiempo”.

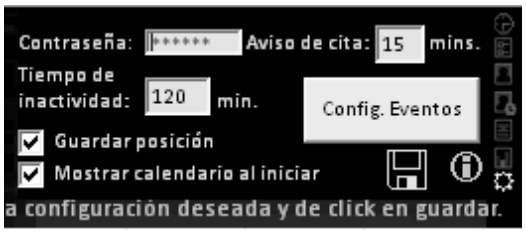
En esta interfaz se mostrará a manera de resumen un listado por días mostrando el total de actividades Pendientes, Trabajadas y Terminadas de las agendas Laboral y Personal, así como el tiempo trabajado e inactivo.

En la parte inferior se muestran los promedios de estas actividades, los cuales se calculan dividiendo la suma de las cantidades diarias respectivas entre el número de días evaluados (los mostrados en la lista). Desde aquí también podrá acceder a la ventana del gráfico de actividades que muestra en un gráfico de barras la información de esta interfaz.



Las opciones de la ventana de Gráficos son: Mostrar Porcentaje o Cantidades, y puede elegir entre graficar el Total de actividades, Actividades Laborales, Actividades Personales y Tiempos.
La gráfica es muy fácil de interpretar.

Interfaz de Configuración



Finalmente, la interfaz de configuración le permitirá realizar algunos sencillos ajustes para el programa en general y sus interfaces.

Las opciones que puede configurar son:
Contraseña: Esta es empleada para cifrar la información almacenada en los archivos, de manera que no puedan ser visualizados por personas no autorizadas. La contraseña puede ser la que el usuario decida, siempre y cuando tenga máximo 8 caracteres, los cuales no se muestran en la pantalla, en su lugar aparece un *.
Aviso de cita: En este campo se escribe el tiempo (en minutos) en el que el usuario desea se le muestre el recordatorio de cita. Una vez que este tiempo llega antes de la cita, la ventana de recordatorio se mostrará cada minuto, a menos que especifique otro mediante los botones de Posponer de dicha ventana.
Tiempo de inactividad: Aquí se escribe el tiempo (en minutos) en que el agente del programa esperará antes de informarle que no ha realizado alguna actividad sugiriéndole algunas opciones (ver el tema del agente).
Guardar posición: si esta casilla está marcada, entonces el programa recordará dónde se ubicaba la ventana principal en la pantalla la última vez que se cerró para aparecer ahí la próxima vez que lo ejecute.
Mostrar calendario al iniciar: si esta casilla está marcada, entonces al iniciar el programa, le mostrará la interfaz principal con el calendario desplegado.
Esta interfaz tiene 3 botones:
Guardar: Este botón guardará cualquier cambio que haya hecho en la configuración. Es importante hacer clic aquí siempre que se realice algún cambio. Los cambios en la configuración son inmediatos, no es necesario reiniciar el programa para que se realicen.
Acerca de: Este botón despliega información del programa, la versión y el nombre del autor directamente en la marquesina.
Config. Eventos: Este botón mostrará la ventana para la configuración de eventos, esta ventana fue descrita anteriormente en la interfaz Reloj / Calendario.

Bases de Datos

Las Bases de Datos se emplean principalmente para administrar grandes cantidades de información y dado que la información que se almacena del y para el programa en este caso no es para nada extensa o compleja, se ha decidido el empleo de varios archivos de texto en su lugar.
Aunque el motivo principal es que el programa está orientado para su uso en una empresa donde el empleo e instalación de software no estándar está muy restringido y eso incluye la instalación y uso de manejadores de bases de datos.
Los archivos están divididos de acuerdo a la sección y tipo de agenda que representan quedando como sigue:
AgendaL.txt – Para los temas de la agenda laboral
AgendaLT.txt – Para los temas terminados de la agenda laboral
AgendaP.txt – Para los temas de la agenda personal
AgendaPT.txt – Para los temas terminados de la agenda personal
ContactosL.txt – Para los contactos laborales
ContactosP.txt – Para los contactos personales
CitasL.txt – Para las citas laborales
CitasP.txt – Para las citas personales
NotasL.txt – Para las notas laborales
NotasP.txt – Para las notas personales
Eventos####.txt – Para los eventos por año
EventosRep.txt – Para los eventos periódicos anuales

Para proteger la información almacenada en estos archivos se utiliza un sencillo algoritmo de encriptación utilizando el método de ### con la contraseña del usuario como llave.

El objetivo es que únicamente el usuario pueda ver y crear sus archivos y aunque sean copiados a otra computadora no sean visibles para alguien ajeno.
Los datos se almacenan y leen de manera secuencial.

Objetos en las interfaces

Para crear una interfaz general, compartida y minimalista, fue necesario crear en una sola ventana todos los elementos necesarios para cada interfaz y agruparlos mediante *Frames*.
Mediante código se reajustan tamaños y posiciones de estos elementos para que sean mostrados correctamente. Esto es sencillo al tenerlos por *Frames*, sin embargo, cada vez que se cambia de interfaz, es necesario reajustar varias propiedades de los objetos tales como visibilidad, posición e incluso datos en los mismos, para que al moverse a través del sistema, los cambios sean transparentes y sin conflictos.
Fue necesario también crear completamente el Calendario para mostrarlo de una manera única pues los objetos de Calendario del mismo Visual Basic no eran armoniosos con el diseño de la interfaz, así como tampoco la funcionalidad. Esto implicó elaborar funciones que crearan el calendario de acuerdo al año o mes a mostrar y que manejaran los eventos a mostrar; también el número de objetos se incrementó considerablemente aunque su manejo fue relativamente sencillo gracias a la posibilidad de crear matrices de objetos y manejarlos únicamente mediante índices.
El reloj implicó el uso de trucos visuales para darle un efecto realista mediante etiquetas, colores y *timers* para el efecto inicial, donde los números se mueven rápidamente desde cero hasta el correspondiente al tiempo actual en horas, minutos y segundos, donde se detiene y continúa incrementándose el segundo a un ritmo normal.
Para mostrar información de una manera dinámica y útil para cada interfaz, el empleo de la marquesina resultó práctico. Se trata de una etiqueta que tiene su propio *timer* para desplazarse de derecha a izquierda y cuya longitud se reajusta cada vez que su contenido cambia. Este tamaño está en función del tipo de letra y tamaño de la misma.
Para moverse entre las interfaces, el uso de una barra de iconos también resultó práctico pues no fue necesario colocar menús que normalmente ocupan mucho espacio y no lucen atractivos. En su lugar esta barra contiene una serie de 7 iconos que se encuentran oscurecidos, exceptuando al de la interfaz actual, que se muestra en colores brillantes para resaltarlo. Con el tiempo el usuario se familiariza con estos iconos, pero al inicio resulta útil el *ToolTipText* que muestra el nombre de la interfaz que representa el icono al dejar el Mouse sobre él unos segundos.
Los iconos se colocaron en un objeto *Image*, que muestra la serie de iconos y que al ser seleccionado se cambia la imagen de acuerdo a la sección de la imagen donde se hizo clic. Las imágenes están almacenadas en un objeto *ImageList*.
El formato de día, mes y semana resalta en tamaño y color el día actual para una rápida visualización. El número de semana se agregó pues en el área de aplicación es un dato muy importante en cuanto a plazos. El efecto de sombras y resaltes se hizo nuevamente con etiquetas y colores grises.
Puesto que la interfaz no tiene una barra de título y no puede minimizarse, era necesario darle al usuario la opción de poder moverla y cerrarla a su antojo.
Para poder moverla se recurrió a las funciones propias de *Windows*® que permiten recolocar la ventana de acuerdo al arrastre del Mouse, que en este caso se hace mediante los números de horas o minutos, únicamente desde la interfaz de Reloj/Calendario y lo cual es indicado al cambiar el cursor al pasarlo sobre estos números.
Para “apagar” o cerrar el programa, se colocó la imagen de un botón de encendido/apagado (que mientras está en “encendido” es de color azul)
Una parte importante a considerar respecto a las interfaces, es que cuando el usuario realiza un cambio en la configuración, este es tomado y mostrado inmediatamente después de guardar la información de configuración, no es necesario reiniciar el programa. Esto es importante por varios motivos, pero principalmente porque la mayoría de las veces se encontrarán varios *timers* en ejecución midiendo tiempos y actividades y la interrupción de los mismos podría ocasionar un resultado erróneo en las mediciones y estadísticas que se explicarán posteriormente.

Además de que al iniciar el programa, se registra la hora de inicio del mismo, dando al usuario un número exacto de uso de la computadora, lo que se vería interrumpido al realizar algún ajuste en la configuración, si tuviera que reiniciarse.
La fase lunar es un elemento casi obligado de cualquier calendario, y aunque el código principal se obtuvo de ecuaciones encontradas en Internet, se realizaron algunos ajustes para la visualización gráfica del porcentaje. Esto fue posible mediante una serie de 8 imágenes de las fases lunares almacenadas en un *ImageList*, y a petición de algunos usuarios, se incluyó la semana santa (pues ayuda a planear las vacaciones que siempre hay por esas fechas).
Este cálculo tiene su base en las fases lunares, y dado que ya se tenía ese dato, fue más sencillo introducirlo en el calendario. Para marcarlas únicamente se hace un cambio en el color de la etiqueta del día del jueves y viernes santos y se agrega al texto mostrado en la etiqueta de la marquesina.
La interfaz de las agendas representó un reto al momento de su diseño, pues era necesario registrar el tiempo empleado en cada actividad con exactitud y de una manera sencilla y amigable para el usuario.
Estamos acostumbrados a los aparatos reproductores de video y sonido, por lo que los botones de Play, Pause y Stop son universalmente conocidos por todos los que en principio utilizarán el programa. Esto permitió crear una interfaz muy simple. Solo es necesario seleccionar el tema y dar clic en Play para iniciar el cronómetro, Pause para detenerlo y Stop para terminarlo.
Los botones son nuevamente etiquetas que mediante fuentes como *Webdings* y *Wingdings* muestran los iconos estándares del triángulo (Play), la doble barra (Pause) y el cuadro (Stop).
El cambio de color al mostrar el botón de pausa se sincronizó con el *timer* del segundero.
En las Agendas, así como en Contactos, Citas y Notas, los datos almacenados se muestran en un objeto *ListView*, con la opción de Reporte. Decidí emplear este objeto por la facilidad de mostrar los datos como una hoja de cálculo con filas y columnas, encabezado y selección múltiple.
Cada interfaz tiene su *ListView* correspondiente, de manera que cuando se cambia de interfaz, se agregan, modifican o almacenan los datos, ya sean temas, contactos, citas, o notas.
Este *ListView* es cargado con la información que le corresponde de los archivos de texto al iniciar el programa.
Al seleccionar algún tema del *ListView*, los datos principales del mismo se concatenan y se muestran en la marquesina.
Para los contactos, citas y notas, se utilizan cajas de texto para cargar y editar la información de los archivos. Utilizando nuevamente los iconos de las fuentes *Webdings* y *Wingdings*, se representan los campos de Nombre, Dirección, Teléfono, Celular y Correo Electrónico, pues además de contribuir al aspecto minimalista de la interfaz no se desperdicia espacio utilizando textos.
Las cajas de texto pueden editarse para agregar un contacto, cita y nota nuevos, para editarlos y para buscar información.
Esta opción de búsqueda, que se encuentra en las mencionadas interfaces, permite al usuario encontrar rápidamente un texto en específico en cualquiera de los campos. La búsqueda se hace de manera dinámica al colocar un “botón” (Etiqueta nuevamente) de Encendido/Apagado de búsqueda, de tal manera que al escribir en cualquiera de los campos, si el botón está en encendido se realiza la búsqueda con ayuda de un *ListView* adicional para mostrar los resultados en la misma ventana, y si está apagado, se entiende que se están capturando datos nuevos.
Para la sección de Estadísticas, los botones mostrados abren cada uno una ventana independiente donde se muestra la información en *ListViews*, cuadros de texto y etiquetas.
En dichas ventanas hay un botón extra que abre otra ventana con un gráfico de barras simple, pero que facilita la visualización de la información mostrada.
Dado que se requería que el programa se ejecutara en cualquier PC sin necesidad de instalaciones ni librerías adicionales, no empleé gráficos prediseñados, sino que opté por la programación de unos sencillos gráficos mediante el uso de formas básicas (rectángulos con efectos de relleno) que únicamente varían sus dimensiones de acuerdo al porcentaje o cantidad que representan.
Esto restringió también el número de series a graficar, así como las opciones disponibles, que se redujeron únicamente a mostrar los 10 programas y ventanas más y menos utilizados, así como también mostrar la información en porcentaje o en tiempo.
Finalmente la interfaz de configuración consta de cajas de texto, etiquetas y C*heckBoxes*. Hay un botón (etiqueta) con el icono de Información que muestra información del autor y del programa en

la marquesina, aprovechando que se tiene esta etiqueta desplegable. Un botón de Configuración de Eventos abre la ventana donde se agregan los eventos al Calendario de manera manual (la manera automática es haciendo clic directamente en el número de día del calendario).
Esta ventana muestra en un listado (*ListBox*), todos los eventos del año seleccionado, y permite agregar, eliminar y editar los eventos haciendo clic en el botón (etiqueta) correspondiente.
También se muestra a manera de *StatusBar* inferior en otra etiqueta, el archivo que está siendo mostrado en ese momento.

Como puede observarse, el uso de etiquetas fue constante en la creación de las interfaces, principalmente motivado por la búsqueda de "botones" minimalistas, que el uso de botones reales no permitía. La flexibilidad de colocar etiquetas transparentes con un icono simulando un botón, es una buena opción si lo que se busca es encajar en los colores y diseños de la interfaz sin verse demasiado "anticuado", pues a pesar de que los botones tienen una opción de estilo gráfico, el hecho de que sean enormes rectángulos que ocupan espacio de más, los hace poco menos que efectivos en una interfaz con muy poco espacio.

Funciones principales

El sistema contiene diversas funciones que permiten interactuar con las interfaces de una manera intuitiva y simple.
A continuación se describen las principales funciones que son el núcleo del sistema y sus interfaces.

Mainform

Dado que prácticamente todas las interfaces comparten botones, fue necesario tener una variable global que almacenara la interfaz activa para poder diferenciar el comportamiento de los botones compartidos, que son:

Agregar
Eliminar
Editar
Guardar

Por lo tanto, la acción clic en cada uno de estos botones identifica primeramente la interfaz y mediante un *Select Case*, realizamos la actividad correspondiente.

La función Agregar, se encarga de crear un nuevo elemento activando campos o mostrando ventanas adicionales según la interfaz.
Para las Agendas, se mostrará una ventana con los campos necesarios y la opción de activar el cronómetro. Para los contactos, citas y notas, al hacer clic en el botón de Agregar, los datos actuales de las cajas de texto se agregan en la lista inferior (*ListView*) y se llama automáticamente a la función de Guardar, para que ésta almacene la información ingresada en el archivo de texto correspondiente.
Como se mencionó anteriormente, el almacenamiento de la información se hace de manera secuencial, según se encuentre la información en el *ListView*.

```
Private Sub BAddAg_Click()
Dim res, LVGenerico As ListView, Texto As String
BAddAg.ForeColor = &HFF00&
 Select Case Interfaz  '
  Case 2:  'Agendas
    If BPersonal.ForeColor = &HE0E0E0 Then
     FNuevoTema.OBPersonal.Value = True
    Else
     FNuevoTema.OBPersonal.Value = False
    End If
    If OBPendientes.Value = True Then
     FNuevoTema.CBStatus.ListIndex = 0
    Else
     FNuevoTema.CBStatus.ListIndex = 1
    End If
    FNuevoTema.Show
  Case 3:  'Contactos
    If Trim(TConNombre.Text) = "" Then MsgBox "Debe escribir el nombre del contacto.",
    If BPersonal.ForeColor = &HE0E0E0 Then  'Personal está activo
     Texto = "Personal"
     Set LVGenerico = LVContactosP
    Else
     Texto = "Laboral"
     Set LVGenerico = LVContactos
    End If
     res = MsgBox("¿Desea agregar los datos capturados a la lista de Contactos " & Text
     If res = vbYes Then
      With LVGenerico.ListItems.Add(, , TConNombre.Text)
       .SubItems(1) = TConCel.Text
```

Para el caso de Eliminar, es necesario que el usuario haya seleccionado previamente un tema del *ListView* para eliminarlo.
Una vez seleccionado el tema, contacto, cita o nota, se da clic en el botón de eliminar y como es de esperarse un mensaje aparece solicitando la confirmación de eliminación del elemento. En caso de elegir “No”, el elemento permanece y no se hace nada.
Si se elige “Sí”, entonces el elemento es eliminado del *ListView* correspondiente y la función de Guardar es llamada automáticamente para almacenar el cambio en el archivo de texto. Esto permite mantener los archivos actualizados constantemente.
Observación: En el caso de las agendas, la opción de selección múltiple (que se explica más adelante) permite elegir más de un tema para trabajarlo, sin embargo al elegirlos, el botón de eliminar se desactiva pues solo es posible eliminar un tema a la vez.

```
Private Sub BDelAg_Click()
Dim res, LVGenerico As ListView, Texto As String, i As Long

 Select Case Interfaz
  Case 2:  'Agendas
           If BPersonal.ForeColor = &HE0E0E0 Then
            If OBPendientes.Value = True Then Set LVGenerico = LVTemasP Else Set LVGe
           Else
            If OBPendientes.Value = True Then Set LVGenerico = LVTemasL Else Set LVGe
           End If
           Texto = "el Tema"
  Case 3:  'Contactos
           If BPersonal.ForeColor = &HE0E0E0 Then
            Set LVGenerico = LVContactosP
           Else
            Set LVGenerico = LVContactos
           End If
           Texto = "el Contacto"
  Case 4:  'Citas
           If BPersonal.ForeColor = &HE0E0E0 Then
            Set LVGenerico = LVCitasP
           Else
            Set LVGenerico = LVCitas
           End If
           Texto = "la Cita"
  Case 5:  'Notas
           If BPersonal.ForeColor = &HE0E0E0 Then
```

La función Editar, permite modificar la información desplegada en el *ListView* y que corresponde a los archivos de texto. Nuevamente para el caso de las Agendas, el proceso es ligeramente distinto al resto.

Para las Agendas, se procede de manera similar a la creación de un tema nuevo, mostrando la ventana de nuevo tema, pero ahora con los datos del tema seleccionado en los campos correspondientes y activos completamente para edición. Una diferencia con la creación de temas, es que el campo de “Tiempo”, ahora tiene un valor que puede ser modificado. Es imperativo que el usuario haya elegido previamente el tema a editar como es de suponerse.

Una vez que los cambios han sido realizados en la ventana de edición, hay dos botones que definen lo que se hará a continuación: Aceptar o Cancelar. El botón Aceptar llevará los cambios realizados a los campos del *ListView* del tema elegido y llamará, como en los casos anteriores, a la función de Guardar que actualizará la información almacenada en el archivo de texto correspondiente. El botón Cancelar, anulará la acción y dejará los datos como se encontraban al hacer clic en ellos, dejando el tema en el que se hizo clic como seleccionado, y por lo tanto, mostrado en la marquesina desplegable.

```
Private Sub BEditAg_Click()
Dim LVGenerico As ListView
Edicion = True
If BAddAg.ForeColor = &H404040 Then
 BAddAg.ForeColor = &HE0E0E0
 BDelAg.ForeColor = &HE0E0E0
 BAddAg.Enabled = True
 BDelAg.Enabled = True
Else
 BAddAg.ForeColor = &H404040
 BDelAg.ForeColor = &H404040
 BAddAg.Enabled = False
 BDelAg.Enabled = False
End If
Select Case Interfaz
Case 2: 'agendas
 If BPersonal.ForeColor = &HE0E0E0 Then
  FNuevoTema.OBPersonal.Value = True
  If OBPendientes.Value = True Then Set LVGenerico = LVTemasP Else Set LVGenerico = LV
 Else
  FNuevoTema.OBPersonal.Value = False
  If OBPendientes.Value = True Then Set LVGenerico = LVTemasL Else Set LVGenerico = LV
 End If

 FNuevoTema.OBPersonal.Enabled = True
 FNuevoTema.OBTrabajo.Enabled = True
 FNuevoTema.Caption = "Editar Tema " & LVGenerico.SelectedItem.Text
 FNuevoTema.TFecha.Enabled = True
 FNuevoTema.ChBActTempo.Visible = False
 FNuevoTema.BRegistrar.Caption = "Aceptar"
```

Para el resto de las interfaces, el hacer clic en el botón de Editar después de haber seleccionado el elemento en el *ListView*, llevará los datos a las cajas de texto de la parte superior según el campo que le pertenezca, y los datos estarán libres para ser editados.
Cuando se hace clic en el botón de Edición, éste cambia de color de un gris a un verde brillante, que le ayuda al usuario a saber que el tema de las cajas de texto está actualmente en edición.
Para guardar los cambios realizados y actualizar el *ListView*, el usuario debe llamar a la función Guardar haciendo clic en el icono del disco. Esto llevará los datos modificados al *ListView*, de ahí al archivo de texto y regresará el color del icono al gris habitual limpiando al mismo tiempo las cajas de texto de la parte superior.

```
 FNuevoTema.IndiceTema = LVGenerico.SelectedItem
 FNuevoTema.Show
Case 3: 'contactos
         If BPersonal.ForeColor = &HE0E0E0 Then
          Set LVGenerico = LVContactosP
         Else
          Set LVGenerico = LVContactos
         End If
         If BAddAg.ForeColor = &H404040 Then
          TConNombre.Text = LVGenerico.SelectedItem.Text
          TConCel.Text = LVGenerico.SelectedItem.ListSubItems(1)
          TConMail.Text = LVGenerico.SelectedItem.ListSubItems(2)
          TConDir.Text = LVGenerico.SelectedItem.ListSubItems(3)
          TConTel.Text = LVGenerico.SelectedItem.ListSubItems(4)
          LVGenerico.Enabled = False
         Else
          TConNombre.Text = ""
          TConCel.Text = ""
          TConMail.Text = ""
          TConDir.Text = ""
          TConTel.Text = ""
          LVGenerico.Enabled = True
         End If
Case 4: 'citas
         If BPersonal.ForeColor = &HE0E0E0 Then
          Set LVGenerico = LVCitasP
         Else
          Set LVGenerico = LVCitas
         End If
         If BAddAg.ForeColor = &H404040 Then
```

La función Guardar tiene que considerar diversos aspectos antes de almacenar la información en los archivos y el más importante de ellos es elegir el archivo correcto.
Los botones que se encuentren activos en el momento de llamar la función serán los que den la pauta para ello.
Primeramente, la función determina con la Variable “Interfaz” el número de interfaz activa, pues como se recordará, todas comparten los mismos botones.
Utilizando un *Select Case*, se procede con las instrucciones adecuadas para cada una de aquellas.

```
Public Sub BGuardarAg_Click()
Dim Archivo, LVGenerico, Texto
BAddAg.ForeColor = &HE0E0E0
BDelAg.ForeColor = &HE0E0E0
BAddAg.Enabled = True
BDelAg.Enabled = True

 Select Case Interfaz  '
  Case 2:
        If BPersonal.ForeColor = &HE0E0E0 Then
         If OBPendientes.Value = True Then
          Archivo = Ruta & "AgendaP.txt"
          Set LVGenerico = LVTemasP
          Texto = "Personal Pendientes"
         Else
          Archivo = Ruta & "AgendaPT.txt"
          Set LVGenerico = LVTemasPT
          Texto = "Personal Terminados"
         End If
        Else
         If OBPendientes.Value = True Then
          Archivo = Ruta & "AgendaL.txt"
          Set LVGenerico = LVTemasL
          Texto = "Laboral Pendientes"
         Else
          Archivo = Ruta & "AgendaLT.txt"
          Set LVGenerico = LVTemasLT
          Texto = "Laboral Terminados"
         End If
```

Posteriormente, para los contactos, citas y notas, solo se tiene dos opciones, o se trata del archivo Personal o el Laboral de cada una de ellas y para saberlo se revisa el estado del botón Personal.
El “botón” es una etiqueta que cambia de color al estar activa a un gris más brillante, cuando está desactivada, el gris es uno oscuro. De esta manera, al revisar el color de la etiqueta podemos definir finalmente si la información se almacenará en el archivo de información Personal o Laboral.

Para enviar la información a los archivos de texto, se abre el archivo (con una variable que al iniciar el programa se ha cargado con la ruta del programa) para operaciones de salida y mediante la instrucción *Write*, se mandan los datos separados por comas. Esta es una manera práctica de manejar los datos de entrada y salida a los archivos pues cuando se leen, es posible asignar cada dato leído por líneas a una variable distinta cada uno.
Una vez enviados todos los elementos del *ListView* mediante un ciclo, el archivo se cierra y se informa al usuario que los datos fueron guardados exitosamente.

```
Case 5:
      If BPersonal.ForeColor = &HE0E0E0 Then
        Archivo = Ruta & "NotasP.txt"
        Set LVGenerico = LVNotasP
        Texto = "Personales"
      Else
        Archivo = Ruta & "NotasL.txt"
        Set LVGenerico = LVNotas
        Texto = "Laborales"
      End If
      If Edicion = True Then
       LVGenerico.SelectedItem.Text = TNfecha.Text
       LVGenerico.SelectedItem.SubItems(1) = TNtema.Text
       Edicion = False
       TNfecha.Text = ""
       TNtema.Text = ""
      End If

      Open Archivo For Output As #1
         For i = 1 To LVGenerico.ListItems.Count
          Write #1, LVGenerico.ListItems(i).Text, LVGenerico.ListItems(i).SubItems(1)
         Next i
      Close #1
      MsgBox "Archivo de Notas " & Texto & " guardado.", vbInformation + vbOKOnly


 End Select
  BDelAg.Enabled = False
  BDelAg.ForeColor = &H404040
  BEditAg.Enabled = False
  BEditAg.ForeColor = &H404040
  LVGenerico.Enabled = True
  BGuardarAg.ForeColor = &HE0E0E0
End Sub
```

Ahora, para las agendas, las opciones no son dos, sino cuatro, pues como se mencionó previamente, además de tener agenda personal y laboral, también se cuentan con temas Pendientes y Terminados.
Realmente el proceso es el mismo mencionado arriba, con la diferencia de que para determinar si se trata de los Pendientes o los Terminados, se cuenta con dos botones de Opción configurados visualmente como botones y no como círculos, situados en la parte inferior de la interfaz de Agendas.
Estos botones tienen dos estados, Verdadero o Falso, y al determinar el estado de uno, automáticamente tenemos el del otro, y sabemos cuál está elegido pues solo puede ser uno de ellos. De esta manera conocemos exactamente qué archivo elegir para almacenar la información correctamente.
El resto del proceso es idéntico al descrito previamente para las otras interfaces.

Para seleccionar los temas Personales o Laborales, hay dos etiquetas con una sola letra (“P” para Personales y “L” para Laborales) que funcionan como botón, y cada una tiene una función para permite conocer cuál está activo. Dado que es una etiqueta funcionando como botón, lo más sencillo fue cambiar el color de la fuente de la etiqueta para indicar que está activa o inactiva.
Y puesto que estas etiquetas son compartidas en las interfaces, al hacer clic en ellas, no solo debe cambiarse el color de su fuente, sino que es necesario verificar la interfaz activa y hacer visibles los controles correspondientes a la misma.
Por ejemplo, si la interfaz activa es la de contactos, pero se muestran los contactos laborales y cambiamos a los personales, es necesario ocultar el *ListView* de los contactos laborales y hacer visible el *ListView* de los personales, y colocar en la marquesina el texto del elemento

seleccionado. Esto se complica un poco al encontrarnos en la interfaz de las agendas, pero se procede de la misma manera, aumentando la verificación de la opción activa para Pendientes y Terminados.

```
Private Sub BPersonal_Click()
 BPersonal.ForeColor = &HE0E0E0
 BTrabajo.ForeColor = &H404040
 Select Case Interfaz
  Case 2:
          If OBPendientes.Value = True Then
            LVTemasL.Visible = False
            LVTemasP.Visible = True
            LVTemasLT.Visible = False
            LVTemasPT.Visible = False
            If LVTemasP.Enabled = True Then LVTemasP.SetFocus
            If LVTemasP.ListItems.Count <> 0 Then
            Lmsg.Caption = LVTemasP.SelectedItem.Text & "-" & LVTemasP.SelectedItem.Sub
                & "-" & LVTemasP.SelectedItem.SubItems(3) & "-" & LVTemasP.SelectedItem
            Else
             Lmsg.Caption = "No hay temas pendientes cargados en la Agenda Personal"
            End If
           Else
            LVTemasL.Visible = False
            LVTemasP.Visible = False
            LVTemasLT.Visible = False
            LVTemasPT.Visible = True
            If LVTemasPT.Enabled = True Then LVTemasPT.SetFocus
            If LVTemasPT.ListItems.Count <> 0 Then
            Lmsg.Caption = LVTemasPT.SelectedItem.Text & "-" & LVTemasPT.SelectedItem.S
                & "-" & LVTemasPT.SelectedItem.SubItems(3) & "-" & LVTemasPT.SelectedIte
            Else
             Lmsg.Caption = "No hay temas terminados cargados en la Agenda Personal"
            End If
           End If
```

El control del cronómetro tiene básicamente 3 funciones ligadas, la del botón *Play*, el botón *Stop* y el Cronómetro en sí, que es la que ejecuta el objeto *Timer*.

El botón *Play* identifica en primera instancia si se trata de la agenda Personal o Laboral, y luego si la vista activa es la de Pendientes o Terminados. Esto es importante, pues si la interfaz activa es la de Pendientes, al presionar *Play* el cronómetro inicia y el tema seleccionado es preparado para recibir el tiempo transcurrido hasta que el botón *Stop* o Pausa son presionados. Mientras tanto su status cambia de “Pendiente” a “En Edición”.
Pero en el caso de que la interfaz activa sea la de Terminados, el tema seleccionado será retirado del listado de Terminados y se agregará a la lista de pendientes. Estos datos, digamos el *ListView* “origen” y el *ListView* “destino”, son almacenados en unas variables genéricas para que una función almacene los datos del *ListView* genérico en el archivo correcto. Al hacer el traspaso de un listado a otro, los datos se guardan en los archivos de texto llamando a la función de Guardar.
También es necesario cambiar los estados de los botones de Terminados y Pendientes.

Después de identificar plenamente la interfaz y los *ListView*s, la imagen del botón debe cambiar y en lugar de mostrar el conocido triángulo de *Play*, debe mostrar ahora la doble barra del símbolo de Pausa, para indicar que el tiempo está siendo medido actualmente para el tema seleccionado y que es posible pausarlo en cualquier momento presionando el mismo botón.
Para hacer más visible el estado de pausa del elemento, el botón cambia de gris claro a un color verde lima y viceversa mediante el uso del *timer* del segundero.
Es necesario también verificar si se encuentra un tema seleccionado, pues el tiempo que mida el *timer* será añadido al tiempo almacenado para ese tema. Sin embargo, es posible usar el *timer* como un simple cronómetro. Si no es encuentra ningún tema seleccionado entonces se le pregunta

al usuario si desea activar el *timer* de todas maneras y si el usuario lo confirma se inicia el conteo, en caso contrario se finaliza este procedimiento.

```
Public Sub BPlayAg_Click()
Dim res, suma, TemSel, PromSel
Dim LVGenerico As ListView
Dim LVGenericoDest As ListView
Dim TermPend As Boolean

 If BPersonal.ForeColor = &HE0E0E0 Then
  If OBPendientes.Value = True Or OpTerActiva = False Then
   Set LVGenerico = LVTemasP
  Else
   res = MsgBox("El Tema o Temas seleccionados serán colocados en la lista de Temas Penc
   If res <> vbYes Then Exit Sub
   Set LVGenerico = LVTemasPT
   Set LVGenericoDest = LVTemasP
   TermPend = True
  End If
 Else
  If OBPendientes.Value = True Or OpTerActiva = False Then
   Set LVGenerico = LVTemasL
  Else
   res = MsgBox("El Tema o Temas seleccionados serán colocados en la lista de Temas Penc
   If res <> vbYes Then Exit Sub
   Set LVGenerico = LVTemasLT
   Set LVGenericoDest = LVTemasL
   TermPend = True
  End If
 End If

If TermPend = True Then
```

Al iniciar el conteo, todos los números (etiquetas) empleados para mostrar normalmente la hora, son reiniciados a “0” y el *timer* llamado “Cronometro” se activa. Este *timer* está configurado para contar segundos y veremos su funcionamiento más adelante.
Para evitar un cambio en el tema seleccionado mientras el cronómetro se encuentra activo, se deshabilitan todos los *ListViews* de las Agendas.

Si el botón ahora de Pausa es presionado se procede de la siguiente manera:
Se regresa el símbolo del triángulo del *Play* al botón así como el *ToolTipText* que estaba como “Pausar Tema/Cronómetro” se regresa a “Continuar Tema/Cronómetro”.
Todas los *ListViews* de las Agendas se habilitan de nuevo. El *timer* “Cronometro” se deshabilita y el tema o temas que estén seleccionados cambian su status de “En Edición” a “Pendiente” nuevamente.

```
If Lmsg.Caption = "No hay tema seleccionado." And LTSecs.Caption = "00" Then
 res = MsgBox("No hay tema seleccionado. ¿Desea activar el cronómetro de cualquier man
 If res = vbNo Then Exit Sub
End If
BPlayAg.Caption = ";"
BPlayAgSh.Caption = ";"
BPlayAg.ToolTipText = "Pausar Tema / Cronómetro"
Cronometro.Enabled = True
If Cron_Reiniciar = False Then
 Cron_Seg = 0
 Cron_Min = 0
 Cron_Hor = 0
End If
Cron_Activo = True
If CuentaRegresiva = False Then
 LTSecs.Caption = "00"
 LMN1.Caption = "0"
 LMN2.Caption = "0"
 LHN1.Caption = "0"
 LHN2.Caption = "0"
 LMG1.Caption = "0"
 LMG2.Caption = "0"
 LHG1.Caption = "0"
 LHG2.Caption = "0"
Else
 LTSecs.Caption = "59"
 LMN1.Caption = Left(CRMinutos, 1)
 LMN2.Caption = Right(CRMinutos, 1)
 LHN1.Caption = Left(CRHoras, 1)
 LHN2.Caption = Right(CRHoras, 1)
 LMG1.Caption = Left(CRMinutos, 1)
 LMG2.Caption = Right(CRMinutos, 1)
 LHG1.Caption = Left(CRHoras, 1)
 LHG2.Caption = Right(CRHoras, 1)
End If

TimerAgente.Enabled = False
ContMin = 0
```

Para hacer el cálculo del tiempo que se añadirá al tema o temas seleccionados, primero se determina cuántos temas se encuentran seleccionados y se almacena el dato en la variable TemSel.
Posteriormente se convierte a segundos el tiempo que marca el cronómetro actualmente y el resultado se divide entre el número de temas seleccionados o TemSel.
A continuación este resultado se suma al valor que cada tema seleccionado del *ListView* tiene en su campo “Tiempo”.

```
suma = 0
suma = Val(LHN1.Caption & LHN2.Caption) * 60
suma = suma + Val(LMN1.Caption & LMN2.Caption)
suma = suma + Val(LTSecs.Caption / 60)

PromSel = suma / TemSel

For i = 1 To LVGenerico.ListItems.Count
 If LVGenerico.ListItems(i).Selected = True Then
  LVGenerico.ListItems(i).SubItems(4) = Round(LVGenerico.ListItems(i).SubItems(4) + Pro
  LVGenerico.ListItems(i).SubItems(9) = Val(LVGenerico.ListItems(i).SubItems(9)) + 1
 End If
Next i
If OBTerminados.Enabled = False Then OBTerminados.Enabled = True
LVGenerico.SetFocus
```

Este cálculo se realiza de esta manera, pues es lo más equilibrado que puede hacerse un estimado del tiempo empleado al realizar varias tareas simultáneamente.
En el caso de que sea solamente un tema el que esté seleccionado, el tiempo del cronómetro pasará íntegro al valor actual.

Existe un *timer* llamado “TimerAgente” que será explicado posteriormente, baste con decir aquí que se trata de un *timer* que mide el tiempo de inactividad del usuario y que al pausar un tema este *timer* es activado.

El botón *Stop* tiene la función de terminar los temas. Esto quiere decir que funcionará únicamente con aquellos temas de la Agenda Personal o Laboral que se encuentren como Pendientes.
Esto es revisado al inicio del procedimiento que lo controla. En caso de que se detecte que el botón Terminados sea el activo, se le informa al usuario mediante un mensaje que el tema ya se encuentra terminado y que si desea activarlo de nuevo deberá presionar el botón *Play*.
La opción de selección múltiple de los *ListViews* está habilitada, por lo tanto uno o varios temas pueden ser terminados al mismo tiempo, y para confirmar que es lo que el usuario desea, se le pide que confirme la acción para continuar.

Dado que el cronómetro puede encontrarse activo al momento de dar clic en este botón, se realiza el mismo cálculo descrito previamente cuando se presiona el botón Pausa, para colocar en los temas seleccionados el tiempo medido en el cronómetro.
A continuación se mueven uno a uno los temas seleccionados del *ListView* de Pendientes activo al *ListView* de Terminados de la Agenda correspondiente. Esto se hace eliminando y agregando los temas de un *ListView* al otro.
Se desactiva el *timer* “Cronómetro”, se reinician los números del *timer* a cero y se llama a la función Guardar dos veces, para almacenar los archivos de las Agendas Pendientes y Terminados.
Finalmente se habilitan nuevamente todos los *ListView* de las Agendas.

```
Private Sub BStopAg_Click()
Dim LVGenerico As ListView, res
Dim LVGenericoDest As ListView


 If BPersonal.ForeColor = &HE0E0E0 Then
  If OBPendientes.Value = True Or OpTerActiva = False Then
   Set LVGenerico = LVTemasP
   Set LVGenericoDest = LVTemasPT
  Else
   MsgBox "El tema ya está terminado. " & vbCrLf & "Puede enviarlo nuevamente a los Tema
   Exit Sub
  End If
 Else
  If OBPendientes.Value = True Or OpTerActiva = False Then
   Set LVGenerico = LVTemasL
   Set LVGenericoDest = LVTemasLT
  Else
   MsgBox "El tema ya está terminado. " & vbCrLf & "Puede enviarlo nuevamente a los Tema
   Exit Sub
  End If
 End If

res = MsgBox("Los temas seleccionados actualmente serán marcados como terminados " & vb(
If res = vbNo Or res = vbCancel Then Exit Sub


 Cron_Reiniciar = True
 BPlayAg.Caption = "4"
 BPlayAgSh.Caption = "4"
```

Timer Cronómetro.

Este *Timer*, se encarga de contabilizar el tiempo cuando el Cronómetro está activo. Está configurado para contar segundos y cada segundo simplemente incrementa los valores de las etiquetas de horas, minutos y segundos de la interfaz de manera coherente. Simplemente al llegar a 60, el segundero se reinicia a cero y se incrementa en uno el minutero. En caso de que el minutero llegue a 60, también se reinicia a cero y se incrementa en uno el indicador de las horas.
Como medida de seguridad, si el indicador de las horas llega a 99, es reiniciado a cero.

```
Private Sub Cronometro_Timer()
 If BPlayAg.ForeColor = &HE0E0E0 Then BPlayAg.ForeColor = vbGreen Else BPlayAg.ForeColor
  Cron_Seg = Cron_Seg + 1
  If Cron_Seg = 60 Then
   Cron_Seg = 0
   Cron_Min = Cron_Min + 1
  End If
  If Cron_Min = 60 Then
   Cron_Min = 0
   Cron_Hor = Cron_Hor + 1
  End If
  If Cron_Activo = False Then Exit Sub
  SgTx = Trim(Str(Cron_Seg))
  If Cron_Seg < 10 Then SgTx = "0" & SgTx
  LTSecs.Caption = SgTx

  If Cron_Min < 10 Then
   LMG1.Caption = "0"
   LMN1.Caption = "0"
   LMG2.Caption = Cron_Min
   LMN2.Caption = Cron_Min
  Else
   LMG1.Caption = Left(Cron_Min, 1)
   LMN1.Caption = Left(Cron_Min, 1)
   LMN2.Caption = Right(Cron_Min, 1)
   LMG2.Caption = Right(Cron_Min, 1)
  End If

  If Cron_Hor < 10 Then
   LHG1.Caption = "0"
   LHN1.Caption = "0"
   LHN2.Caption = Cron_Hor
   LHG2.Caption = Cron_Hor
  Else
   LHG1.Caption = Left(Cron_Hor, 1)
   LHN1.Caption = Left(Cron_Hor, 1)
```

Load.

Una de las funciones o procedimientos más comunes e importantes en todo programa, es la inicialización de la interfaz, con sus ventanas y variables listas para funcionar.
Esta no es la excepción, y la función *Load* (carga) es que la pone el sistema a punto para trabajar. Se describe a continuación paso a paso.
Los datos de la configuración del programa, Contraseña, Calendario activo, Posición (x,y,), Aviso de cita y Tiempo de inactividad, son almacenados en el registro de *Windows*®. Entonces, como paso inicial de la función de carga, se leen estos datos del registro para almacenarlos en los campos de texto de la interfaz de configuración, excepto por las coordenadas de la posición inicial de la interfaz que después de leer esta información y verificar que la opción guardar Posición esté activa, entonces son usadas para las propiedades *Left* y *Top* de la ventana principal (MainForm). En caso de que la opción no esté activa, entonces la posición inicial de la ventana será (0,0).

```
Private Sub Form_Load()
 Dim DiaBase, DiasAnt, pos, i, LVGenerico As ListView
 Dim XMain, YMain, Arch
 On Error GoTo ErrorArchivo
 TPsswd.Text = GetSetting("iRobot", "Config", "Psswd", "iRobot")
 ChBMostCal.Value = Val(GetSetting("iRobot", "Config", "CalOn", "1"))
 TBStandBy.Text = GetSetting("iRobot", "Config", "StandBy", "10")
 ChBGuardarPos.Value = GetSetting("iRobot", "Config", "SavePos", "1")
 TBAvisoCita.Text = GetSetting("iRobot", "Config", "DateRecall", "15")
 XMain = Val(GetSetting("iRobot", "Config", "XPos", "0"))
 YMain = Val(GetSetting("iRobot", "Config", "YPos", "0"))

 If ChBGuardarPos.Value = 1 Then
  MainForm.Left = YMain
  MainForm.Top = XMain
 Else
  MainForm.Left = 0
  MainForm.Top = 0
 End If

 ContMin = 0

 Set m_UtilVentanas = New cWindows

 MainForm.Height = 3675
 MainForm.Width = 3915
 FrCitas.Left = 50
 FrCitas.Top = 45
 FrNotas.Left = 50
 FrNotas.Top = 240
 FrContactos.Left = 0
 FrContactos.Top = 50
 FrGraficos.Left = 0
 FrGraficos.Top = 0
```

Ya que la construcción de la interfaz se realizó por medio de *Frames* ubicados en toda la ventana para facilitar su edición, es necesario que éstos sean colocados en su posición real al iniciar el programa. Así que después de definir el tamaño de la interfaz, 3675 x 3915 pixeles, los *Frames* son ubicados uno a uno en su lugar correcto mediante sus propiedades *Left* y *Top* que serán relativos a la esquina superior izquierda de la ventana principal.

Uno de los datos que son mostrados en las estadísticas, es la hora en la que se inició el programa, así que este dato es leído en este momento y almacenado en una variable global llamada "HoraInicio". Las variables para los números de los segundos, minutos y horas son inicializadas en cero y las etiquetas de Mes, día, día de la semana (mediante la función "*Datepart*") cargadas con la información correspondiente a la fecha actual.

A continuación se llaman tres funciones que serán descritas posteriormente en las funciones secundarias, CalculaSemanaSanta, CreaCal y MuestraFaseLunar.
La primera devolverá el domingo de Resurrección de acuerdo a la fe católica y al año en curso. Esto es para poder marcar en el calendario los días Jueves y Viernes santos que normalmente son días de asueto en nuestro calendario laboral.
La segunda función, crea el calendario de acuerdo día, mes y año solicitado. Esto es, carga la numeración correcta en las etiquetas de días en el calendario, y marca con los colores adecuados los días, de acuerdo a si ya pasaron, si están por pasar, si tienen evento y si es el día actual.
La tercera, como su nombre lo indica, mostrará una imagen de la fase lunar de la fecha recibida como parámetro.

La siguiente etapa, es preparar cada una de las interfaces del sistema, principalmente respecto a la información almacenada en los archivos que será mostrada en los *ListViews*.
Para hacerlo se avanza por secciones, primero las agendas, luego contactos, citas, notas y finalmente las estadísticas.
Se abren los archivos, se limpian los *ListViews* y se llenan con los datos de los archivos mediante ciclos, leyendo en variables (V0, V1, ..., V9) las columnas de los mismos.

```
Marquesina = Lmsg.Caption
If ChBMostCal.Value = 0 Then Call LDisplay_Click
PLPhase.Top = 3100
PLPhase.Left = 3300
MuestraFaseLunar Day(Date), Month(Date), Year(Date)

'********************Prepara LVTemasL
    For i = 1 To 4
     Select Case i
     Case 1:
      Nombre = Ruta & "AgendaL.txt"
      Set LVGenerico = LVTemasL
     Case 2:
      Nombre = Ruta & "AgendaP.txt"
      Set LVGenerico = LVTemasP
     Case 3:
      Nombre = Ruta & "AgendaLT.txt"
      Set LVGenerico = LVTemasLT
     Case 4:
      Nombre = Ruta & "AgendaPT.txt"
      Set LVGenerico = LVTemasPT
     End Select
     LVGenerico.ListItems.Clear
     LVGenerico.HideSelection = False
     cad = Dir(Nombre)
     If cad <> "" Then
        Open Nombre For Input As #1
        Fila = 0
        Do While Not EOF(1)
         Input #1, v0, v1, v2, v3, v4, v5, v6, v7, v8, v9

           With LVGenerico.ListItems.Add(1, , Val(v0)) 'ID
            .SubItems(1) = v1 'Tema
            .SubItems(2) = v2  'Clasificación
            .SubItems(3) = v3  'Actividad
            .SubItems(4) = Val(v4)  'Tiempo
            .SubItems(5) = v5  'Fecha
```

Para el caso de las estadísticas, el archivo que se abre contiene el resumen de los tiempos y cantidad de temas trabajados los últimos días y al iniciar el programa, es agregada una línea que corresponde al día actual donde se almacenarán los datos al finalizar el programa. Al crear la línea, todos los campos son inicializados en cero.

Finalmente es necesario hacer un llamado a una función, creada con las funciones propias de *Windows*® que permite “escanear” las ventanas que se encuentran abiertas y poder clasificarlas y guardarlas en un objeto *ListView* llamado LVVentanas para las operaciones del Agente. Esto es explicado posteriormente en la sección de Clase.

```
      Do While Not EOF(1)
             Input #1, v0, v1, v2, v3, v4, v5, v6, v7, v8, v9
               With FEstAct.LVResumen.ListItems.Add(1, , Format(v0, "mm-dd-yyyy")) 'Fec
                .SubItems(1) = v1  'L pendientes
                .SubItems(2) = v2  'L trabajados
                .SubItems(3) = v3  'L terminados
                .SubItems(4) = v4  'L Tiempo trabajado
                .SubItems(5) = v5  'P Pendientes
                .SubItems(6) = v6  'P trabajados
                .SubItems(7) = v7  'P terminados
                .SubItems(8) = v8  'P Tiempo trabajado
                .SubItems(9) = v9  'Tiempo inactivo
               End With
            Loop

    Close #1
End If
UltReg = FEstAct.LVResumen.ListItems.Count + 1
With FEstAct.LVResumen.ListItems.Add(, , Format(Date, "mm-dd-yyyy"))
 .SubItems(1) = LVTemasL.ListItems.Count
 .SubItems(2) = "0"
 .SubItems(3) = "0"
 .SubItems(4) = "0"
 .SubItems(5) = LVTemasP.ListItems.Count
 .SubItems(6) = "0"
 .SubItems(7) = "0"
 .SubItems(8) = "0"
 .SubItems(9) = "0"
End With
FEstAct.LVResumen.Refresh
CargaInicial = True
Call m_UtilVentanas.EnumTopWindows(LVVentanas)
CargaInicial = False
Exit Sub
ErrorArchivo:
MsgBox "Hubo un error al ubicar los archivos. " & vbCrLf & "Es posible que la ruta '" &
End
```

Así, una vez colocados los valores iniciales en todos los campos de las interfaces, el programa está listo para trabajar.

Unload

Como nota adicional, existe un evento contrario al de carga, que es el de descarga, el cual se ejecuta previamente a terminar por completo el programa y cuando éste es cerrado sorpresivamente, es decir, sin utilizar la función (o botón en este caso) propia para este fin.
Es posible colocar funciones que se realicen antes de cerrar y esto es útil para almacenar la información de último momento y en este procedimiento se hace la llamada a la función GuardaEst, que almacena las estadísticas recopiladas en el día y que es la única información que debe almacenarse de forma “automática” pues no hay un botón específico para hacerlo.
Esta función es descrita posteriormente. Así también, en el caso de que la opción de guardar la ubicación de la ventana principal esté activada en la configuración, estos datos se almacenan en el registro de *Windows*®.

ImOnOff

Existe un botón en la interfaz que “apaga” el programa, y al hacer clic en él se realiza exactamente lo descrito en el párrafo anterior, almacenando la información de ubicación y los datos de la estadística. Adicional a esto, la función contiene la instrucción “*End*” que termina completamente el programa.

```
Private Sub ImOnOff_Click()
 If ChBGuardarPos.Value = 1 Then
  SaveSetting "iRobot", "Config", "XPos", MainForm.Top
  SaveSetting "iRobot", "Config", "YPos", MainForm.Left
 End If
 GuardaEst
 End
End Sub
```

Para agregar eventos, existe una opción específica en la configuración. Sin embargo, para facilitar al usuario esta tarea y pensando en lo intuitivo del programa, la lógica nos llevaría a pensar que al hacer doble clic en algún día del calendario, debería pasar algo. Así que se implementó la opción de que la hacer doble clic, se detectan el día y el mes y se abre la ventana de carga de eventos ya con los datos de la fecha seleccionados, de manera que solo hará falta describir el evento y marcar la opción de “Repetir anualmente” en caso necesario para agregar el evento muy fácilmente.

```
Private Sub LCalD_DblClick(Index As Integer)
 If LCalD(Index).ForeColor <> &H202020 Then
  Select Case Left(UCase(LCalM.Caption), 3)
   Case "ENE": FConfig.mes = 1
   Case "FEB": FConfig.mes = 2
   Case "MAR": FConfig.mes = 3
   Case "ABR": FConfig.mes = 4
   Case "MAY": FConfig.mes = 5
   Case "JUN": FConfig.mes = 6
   Case "JUL": FConfig.mes = 7
   Case "AGO": FConfig.mes = 8
   Case "SEP": FConfig.mes = 9
   Case "OCT": FConfig.mes = 10
   Case "NOV": FConfig.mes = 11
   Case "DIC": FConfig.mes = 12
  End Select
  FConfig.dia = LCalD(Index).Caption
  FConfig.DesdeCal = True
  FConfig.LAdd_Click
 End If
End Sub
```

Cuando se “navega” en el calendario avanzando o retrocediendo, se hizo evidente la necesidad de agregar una manera rápida de regresar a la fecha actual para no tener que ir de mes en mes hasta la fecha actual. Lo más obvio fue asignarle esta función a la etiqueta de día. Así que al hacer doble clic en ella, se llaman las funciones que calcula la semana santa y que crea el calendario de acuerdo a la fecha actual, mostrando así el calendario del mes actual.

```
Private Sub LDia_DblClick()
 DomRES = CalculaSemanaSanta(Year(Date))
 CreaCal Day(Date), Month(Date), Year(Date), DomRES
End Sub
```

Para guardar los datos en el registro de *Windows*®, se hizo una función que es llamada al dar clic en el icono de disco (Guardar) en la interfaz de configuración. La función almacenará en el registro los siguientes datos:

Psswd – La contraseña utilizada para encriptar los datos de los archivos.
CalOn – (1/0) para almacenar la opción para tener visible el calendario al momento de iniciar el programa.
StandBy – Tiempo de inactividad que el Agente del programa esperará antes de mostrar un aviso de Inactividad con sugerencias de temas a trabajar.
SavePos –(1/0) para determinar si la posición actual de la ventana principal en el escritorio de *Windows*® será almacenada o no para mostrarla ahí cuando se abra nuevamente el programa.
DateRecall – Tiempo en que el programa mostrará por default el aviso de una Cita.

```
Private Sub LGuardarCfg_Click()
If Val(TBStandBy.Text) = 0 Then MsgBox "El valor del tiempo de espera debe ser mayor a
 SaveSetting "iRobot", "Config", "Psswd", TPsswd.Text
 SaveSetting "iRobot", "Config", "CalOn", ChBMostCal.Value
 SaveSetting "iRobot", "Config", "StandBy", TBStandBy.Text
 SaveSetting "iRobot", "Config", "SavePos", ChBGuardarPos.Value
 SaveSetting "iRobot", "Config", "DateRecall", TBAvisoCita.Text
 MsgBox "La configuración se guardó correctamente.", vbInformation + vbOKOnly, "Configu
 LGuardarCfg.ForeColor = &H00E0E0
End Sub
```

Todos estos datos pueden ser modificados en la interfaz de configuración, desde donde también se guardan.

Form_MouseDown

Para poder mover la ventana por el monitor, tratándose de una ventana sin barra de título, es necesario utilizar las funciones de *Windows*®. Cuando se presiona el botón del ratón sin soltarlo, sobre las etiquetas que muestran la hora, se dispara una función llamada *ReleaseCapture*[8] y otra llamada *SendMessage* con los datos de la ventana.

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Sin
    'Para mover una ventana sin barra de título

    ReleaseCapture
    SendMessage hwnd, WM_NCLBUTTONDOWN, HTCAPTION, ByVal 0&
End Sub
```

ReleaseCapture
Libera la captura del *Mouse* de una ventana en el hilo actual y restaura el procesamiento normal de entrada del *Mouse*. Una ventana que ha capturado el *Mouse*, recibe todas las entradas sin importar la posición del cursor, excepto cuando algún botón del *Mouse* es presionado mientras el cursor se encuentra señalando otra ventana de otro hilo.

Esta función es llamada para poder controlar las entradas del *Mouse* sin que interfieran los datos de alguna otra ventana, es decir, prepara al operativo para la siguiente función:

Sendmessage
Envía un mensaje específico a una ventana o ventanas. Esta función llama el procedimiento de la ventana a la ventana específica y no regresa hasta que el procedimiento de la ventana ha procesado el mensaje.
Como parámetros tiene:
El manejador de la ventana: hwnd, que indica la ventana que recibirá el procedimiento.
wMsg, el mensaje a ser enviado (en este caso el mensaje es WM_NCLBUTTONDOWN[9], que es colocado cuando el usuario presiona el botón izquierdo del *Mouse* mientras el cursor se encuentra dentro de un área no cliente de una ventana. El mensaje es enviado a la ventana que contiene el cursor.

wParam que envía HTCaption=2 y que corresponde al título de la ventana. Esto es resultado del procesamiento de WM_NCHITTEST[10] que es enviado a una ventana con el fin de determinar qué parte de la ventana corresponde a una coordenada particular de la pantalla.
Es decir, al hacer clic en un área no utilizada de la ventana para mover la misma, el movimiento es redirigido a la barra de título de la ventana.

Las interfaces de Contactos, Citas y Notas, tienen una función de búsqueda que permite localizar rápidamente dichos elementos mediante filtros combinados en tiempo real. Es decir, comienza a buscar coincidencias al momento que se comienza a escribir en cualquiera de los campos.
Para que este método funcione, era necesario diferenciar el estado en el que los campos muestran información, sirven para capturar o editar datos y finalmente para cuestiones de búsqueda.
Para esta última en particular, se colocó un botón de Encendido / Apagado de la función de búsqueda mediante un icono. Al hacer clic en él, la función se activa, al hacer clic en él nuevamente la función se apaga.
También se necesitaría mostrar dinámicamente los resultados, para lo cual, la opción más simple, fue colocar un *ListView* adicional de búsqueda, donde las coincidencias al momento de buscar, se irían colocando o eliminando de acuerdo a los criterios. Para esto, al seleccionar el usuario el modo de búsqueda, primero se coloca el *ListView* de Búsqueda en las mismas coordenadas del *ListView* de la interfaz y con los mismos títulos de las columnas. Los datos, en caso de haber, se borran de la búsqueda, el icono (etiqueta) se cambia de color para indicar que se encuentra activo y el *ToolTipText* también se cambia para informar al usuario el estado del mismo.

```
Private Sub LModo_Click()
 If LModo.ForeColor = &H00E0E0E0 Then
  LModo.ForeColor = &HFF00&
  LModo.ToolTipText = "Modo Buscar activo"
  LNuevaNota.Caption = "Buscar Nota:"
 Else
  LModo.ForeColor = &H00E0E0E0
  LModo.ToolTipText = "Activar Modo Buscar"
  LNuevaNota.Caption = "Nueva Nota:"
 End If

 If LModoC.ForeColor = &H00E0E0E0 Then
  LModoC.ForeColor = &HFF00&
  LModoC.ToolTipText = "Modo Buscar activo"
  LVBusqueda.Width = LVNotas.Width
  LVBusqueda.Height = LVNotas.Height
  LVBusqueda.Top = LVNotas.Top + 1700
  LVBusqueda.Left = LVNotas.Left
  LVBusqueda.ColumnHeaders.Clear
  For i = 1 To LVNotas.ColumnHeaders.Count
   col = LVNotas.ColumnHeaders(i).Text
   LVBusqueda.ColumnHeaders.Add i, , LVNotas.ColumnHeaders(i).Text
  Next i
  LVBusqueda.Visible = True
 Else
  LModoC.ForeColor = &H00E0E0E0
  LModoC.ToolTipText = "Activar Modo Buscar"
  LVBusqueda.Visible = False
 End If

End Sub
```

Esto se hace con los objetos correspondientes de las otras interfaces.

NextM, PrevM

Para avanzar o retroceder en los calendarios, simplemente se llama la función que los crea, colocando un +1 o -1 en el mes mostrado según corresponda. Es lógico deducir que debe haber un control para cuando el mes llega al 12 o al 1, en cuyos casos los meses son reiniciados a 1 o a 12, y el año aumentado o disminuido en uno respectivamente.

```
Private Sub LNextM_Click()
 If MesCalAct + 1 = 13 Then MesCalAct = 0: AñoCalAct = AñoCalAct + 1
 CreaCal 1, MesCalAct + 1, AñoCalAct, CalculaSemanaSanta(AñoCalAct)
End Sub

Private Sub LNextM_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
 LNextM.ForeColor = &HFF00&
End Sub

Private Sub LNextM_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
 LNextM.ForeColor = &HC0C0C0
End Sub

Private Sub LPrevM_Click()
 If MesCalAct - 1 = 0 Then MesCalAct = 13: AñoCalAct = AñoCalAct - 1
 CreaCal 1, MesCalAct - 1, AñoCalAct, CalculaSemanaSanta(AñoCalAct)
End Sub

Private Sub LPrevM_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
 LPrevM.ForeColor = &HFF00&
End Sub

Private Sub LPrevM_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
 LPrevM.ForeColor = &HC0C0C0
End Sub
```

ListView click

Cuando el usuario hace clic en algún elemento del *ListView* de cualquier interfaz, los datos del mismo deben ser mostrados en la marquesina, pues es uno de los usos de ésta.

Para hacerlo, cuando se detecta el clic en el elemento, se concatenan los campos más importantes del elemento y se asigna al texto de la marquesina: *Lmsg*.*Caption*. Así mismo, es necesario activar los botones de Edición y Eliminación, que normalmente se encuentran desactivados.

Para el caso de la interfaz de agendas, debido a que pueden ser seleccionados dos o más elementos, se requiere un tratamiento especial.

Cuando el usuario hace clic en el *ListView* de la Agenda, ya sea Laboral o Personal, se busca en la lista completa y cada vez que se encuentra un elemento seleccionado, entonces comienza a concatenarse en el campo *Caption* de la marquesina. Al mismo tiempo se contabiliza el número de elementos seleccionados pues si es mayor de uno, entonces los botones de Editar y Eliminar se mantienen desactivados (no es posible editar o eliminar más de un elemento al mismo tiempo), en caso contrario deben activarse.

```
Private Sub LVCitas_Click()
If LVCitas.ListItems.Count = 0 Then Exit Sub
    If LVCitas.SelectedItem.Index <> -1 Then Lmsg.Caption = LVCitas.SelectedItem.Text & _
    "-" & LVCitas.SelectedItem.SubItems(1) & "-" & LVCitas.SelectedItem.SubItems(2) _
    & "-" & LVCitas.SelectedItem.SubItems(3) & "-" & LVCitas.SelectedItem.SubItems(4) & "|"

  BDelAg.Enabled = True
  BDelAg.ForeColor = &H0E0E0E0
  BEditAg.Enabled = True
  BEditAg.ForeColor = &H0E0E0E0
End Sub
```

Pendientes / Terminados

Existen dos botones en la parte inferior de las Agendas que son del tipo *OptionButton* con el estilo Gráfico activado, de manera que solo uno de los dos puede estar presionado a la vez. Estos botones intercambian la información que se muestra en la interfaz de Pendientes a Terminados y viceversa. El cambio en el código consiste en mostrar el *ListView* adecuado, ocultar los otros, y concatenar los datos de los temas seleccionados para mostrarlos en la marquesina. En el caso de que el Cronómetro esté activo, estas opciones no están disponibles, es necesario pausarlo previamente para evitar inconsistencias en el cálculo del tiempo en el tema correcto.
Adicional a esto, si al cambiar la vista, la nueva no contiene temas aún, la marquesina tendrá un texto informándolo.

```
Private Sub OBPendientes_Click()
If OBPendientes.Value = True Then
 OBTerminados.ForeColor = &H0&
 OBPendientes.ForeColor = &HFFFFFF
End If
If BPersonal.ForeColor = &HE0E0E0 Then
 LVTemasP.Visible = True
 LVTemasL.Visible = False
 LVTemasPT.Visible = False
 LVTemasLT.Visible = False
 If Cronometro.Enabled = True Then Exit Sub
 If LVTemasP.ListItems.Count = 0 Then Lmsg.Caption = "No hay temas pendientes cargados en la
 'MsgBox LVTemasP.SelectedItem.Index
 If LVTemasP.SelectedItem.Index <> -1 Then Lmsg.Caption = LVTemasP.SelectedItem.Text & _
 "-" & LVTemasP.SelectedItem.SubItems(1) & "-" & LVTemasP.SelectedItem.SubItems(2) _
 & "-" & LVTemasP.SelectedItem.SubItems(3) & "-" & LVTemasP.SelectedItem.SubItems(4) & "|"
 LVTemasP.ListItems(LVTemasP.SelectedItem.Index).Selected = True
 If LVTemasP.Enabled = True Then LVTemasP.SetFocus
Else
 LVTemasL.Visible = True
 LVTemasP.Visible = False
 LVTemasLT.Visible = False
 LVTemasPT.Visible = False
 If Cronometro.Enabled = True Then Exit Sub
 If LVTemasL.ListItems.Count = 0 Then Lmsg.Caption = "No hay temas pendientes cargados en la
 'MsgBox LVTemasL.SelectedItem.Index
 If LVTemasL.SelectedItem.Index <> -1 Then Lmsg.Caption = LVTemasL.SelectedItem.Text & _
 "-" & LVTemasL.SelectedItem.SubItems(1) & "-" & LVTemasL.SelectedItem.SubItems(2) _
 & "-" & LVTemasL.SelectedItem.SubItems(3) & "-" & LVTemasL.SelectedItem.SubItems(4) & "|"
 LVTemasL.ListItems(LVTemasL.SelectedItem.Index).Selected = True
 If LVTemasL.Enabled = True Then LVTemasL.SetFocus
End If

End Sub
```

Barra de Iconos

Dado que la interfaz ha sido creada de manera visual más que práctica, la mayoría de los botones en la misma o son etiquetas con tipografías simbólicas o son contenedores de imágenes.
Para el caso de la barra del menú o barra de iconos se creó una imagen alargada con los siete iconos que representan cada una de las interfaces del programa, incluyendo la configuración.
Para atacar el problema de activar y desactivar los “botones” de esta barra, la imagen fue editada para resaltar cada uno de los iconos mientras se oscurecía el resto y guardando entonces 7 imágenes distintas que colocadas en un *ImageList* pueden ser llamadas fácilmente mediante un índice.
Este índice se determina de acuerdo a las coordenadas del clic sobre el objeto *Picture* que contiene los iconos, que dio el usuario para visualizar la interfaz.
Dado que los iconos están colocados de manera vertical en la imagen, entonces solo nos interesa la coordenada Y del clic, y mediante un *Select Case*, podemos saber exactamente el icono “presionado” y ejecutar las acciones correspondientes.

En todas ellas lo que se hace primeramente es asignar a una variable global llamada Interfaz, el número de la interfaz relacionada al icono (es útil en casos que después se explicarán).
Se coloca la imagen de iconos adecuada (con el icono seleccionado en color brillante y el resto oscurecido) tomada del *ImageList*. El *ToolTipText* es cargado con el título de la interfaz, la marquesina se carga con información también y es redimensionada según el contenido. Y principalmente se muestran y ocultan *Frames*, etiquetas y botones según la interfaz.
Por ejemplo, para mostrar el calendario, todos los *Frames* se ocultan (pues las etiquetas de días en el calendario son las únicas que se encuentran en el lugar correcto de la interfaz desde el diseño) y se muestra el botón de apagado. En cambio, para los contactos, se muestran el *Frame* de contactos y el *Frame* de botones (Agregar, eliminar, editar y guardar) y el resto de *Frames* se mantienen ocultos. El botón de apagado también se oculta. Además de que un fondo oscuro se coloca sobre las etiquetas de los días del calendario para ocultarlos, lo mismo sucede con el resto de interfaces.
Adicionalmente a estos cambios generales, es necesario realizar algunos ajustes por interfaz.

```
Public Sub PIconos_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
 Select Case Y
   Case 0 To 180:    'Número Interfaz para Reloj / Calendario
                     Interfaz = 1
                     PIconos.Picture = ImageIcons.ListImages(1).Picture
                     PIconos.ToolTipText = "Reloj/Calendario"
                     Lmsg.Caption = Marquesina
                     Lmsg.Width = Len(Lmsg.Caption) * 90
                     If LDisplay.Caption = "5" Then
                      MainForm.Height = 3675
                     End If
                     '************************************Muestra
                     ImOnOff.Visible = True
                     Timer1.Enabled = True
                     '************************************Oculta
                     'FBotonesAg.Visible = False
                     IFondoC.Visible = False
                     BPlayAg.Visible = False
                     BPlayAgSh.Visible = False
                     BStopAg.Visible = False
                     BStopAgSh.Visible = False
                     OBPendientes.Visible = False
                     OBTerminados.Visible = False
                     FrAgenda.Visible = False
                     FrBotones.Visible = False
                     FrCitas.Visible = False
                     FrCitas2.Visible = False
                     FrNotas.Visible = False
                     FrNotas2.Visible = False
                     FrConfig.Visible = False
                     FrGraficos.Visible = False
                     FrContactos.Visible = False
                     FrContactos2.Visible = False
                     Cron_Activo = False


   Case 181 To 370:  'Número Interfaz para Agendas
```

En las Agendas, los números que sirven para mostrar la hora en el calendario se utilizan para desplegar el cronómetro, por lo tanto cuando se cambia de cualquier interfaz a la de las Agendas, a las etiquetas de los números se les asigna el valor de unas variables globales (Cron_Seg, Cron_Min, Cron_Hor) que contienen los últimos valores que tuvo el cronómetro, pues éste podría estar ejecutándose al momento de cambiar de interfaz y al regresar, estos valores deben ser mostrados para continuar el conteo. Se le hacen también unos ajustes para un correcto despliegue, ya que se tienen que agregan ceros a la izquierda cuando el número de horas, minutos o segundos son menores a 10.

```
SgTx = Cron_Seg
If Cron_Seg < 10 Then SgTx = "0" & SgTx
LTSecs.Caption = SgTx
If Cron_Min < 10 Then
 LMG1.Caption = "0"
 LMN1.Caption = "0"
 LMG2.Caption = Cron_Min
 LMN2.Caption = Cron_Min
Else
 LMG1.Caption = Left(Cron_Min, 1)
 LMN1.Caption = Left(Cron_Min, 1)
 LMN2.Caption = Right(Cron_Min, 1)
 LMG2.Caption = Right(Cron_Min, 1)
End If

If Cron_Hor < 10 Then
 LHG1.Caption = "0"
 LHN1.Caption = "0"
 LHN2.Caption = Cron_Hor
 LHG2.Caption = Cron_Hor
Else
 LHG1.Caption = Left(Cron_Hor, 1)
 LHN1.Caption = Left(Cron_Hor, 1)
 LHN2.Caption = Right(Cron_Hor, 1)
 LHG2.Caption = Right(Cron_Hor, 1)
End If
```

También los *ListViews* deben ser actualizados con la información actual según esté activo el botón Laboral o Personal y Terminados o Pendientes, esta actualización incluye la visibilidad y foco del *ListView* correspondiente y la concatenación de temas seleccionados para mostrarlos en la marquesina. En caso de que no haya ningún tema en el *ListView*, el mensaje desplegado será que no hay tema en la agenda seleccionada.
De manera similar se trabajan los *ListViews* de las interfaces de Contactos, Citas y Notas. La diferencia es que en las Citas y Notas se utiliza la función Date y Time para mostrar Fecha y Hora en los campos de las interfaces destinados para esto, pues en dichas interfaces esa información se muestra automáticamente.

```
If BPersonal.ForeColor = &HE0E0E0 And OBPendientes.Value = True Then
 LVTemasP.Visible = True
 LVTemasL.Visible = False
 LVTemasPT.Visible = False
 LVTemasLT.Visible = False
 If LVTemasP.ListItems.Count = 0 Then Lmsg.Caption = "No hay temas pendientes cargados en la
 If LVTemasP.SelectedItem.Index <> -1 Then Lmsg.Caption = LVTemasP.SelectedItem.Text & _
 "-" & LVTemasP.SelectedItem.SubItems(1) & "-" & LVTemasP.SelectedItem.SubItems(2) _
 & "-" & LVTemasP.SelectedItem.SubItems(3) & "-" & LVTemasP.SelectedItem.SubItems(4) & "|"
 LVTemasP.ListItems(LVTemasP.SelectedItem.Index).Selected = True
 If LVTemasP.Enabled = True Then LVTemasP.SetFocus
ElseIf BPersonal.ForeColor = &HE0E0E0 And OBPendientes.Value = False Then
 LVTemasP.Visible = False
 LVTemasL.Visible = False
 LVTemasPT.Visible = True
 LVTemasLT.Visible = False
 If LVTemasPT.ListItems.Count = 0 Then Lmsg.Caption = "No hay temas terminados cargados en la
 If LVTemasPT.SelectedItem.Index <> -1 Then Lmsg.Caption = LVTemasPT.SelectedItem.Text & _
 "-" & LVTemasPT.SelectedItem.SubItems(1) & "-" & LVTemasPT.SelectedItem.SubItems(2) _
 & "-" & LVTemasPT.SelectedItem.SubItems(3) & "-" & LVTemasPT.SelectedItem.SubItems(4) & "|"
 LVTemasPT.ListItems(LVTemasPT.SelectedItem.Index).Selected = True
 If LVTemasPT.Enabled = True Then LVTemasPT.SetFocus
ElseIf BPersonal.ForeColor = &H404040 And OBPendientes.Value = True Then
 LVTemasL.Visible = True
 LVTemasP.Visible = False
 LVTemasLT.Visible = False
 LVTemasPT.Visible = False
 If LVTemasL.ListItems.Count = 0 Then Lmsg.Caption = "No hay temas pendientes cargados en la
 If LVTemasL.SelectedItem.Index <> -1 Then Lmsg.Caption = LVTemasL.SelectedItem.Text & _
 "-" & LVTemasL.SelectedItem.SubItems(1) & "-" & LVTemasL.SelectedItem.SubItems(2) _
 & "-" & LVTemasL.SelectedItem.SubItems(3) & "-" & LVTemasL.SelectedItem.SubItems(4) & "|"
 LVTemasL.ListItems(LVTemasL.SelectedItem.Index).Selected = True
If LVTemasL.Enabled = True Then LVTemasL.SetFocus
ElseIf BPersonal.ForeColor = &H404040 And OBPendientes.Value = False Then
```

Para el caso de los Reportes, dado que la interfaz es muy sencilla y únicamente contiene dos botones grandes que dan acceso a las ventanas de estadísticas, solo es necesario colocar el fondo negro que cubre cualquier interfaz previa, se muestra el *Frame* con los botones y se oculta el resto, y algo no menos importante, el tamaño de la ventana se reduce al tamaño mínimo (el que tiene la ventana principal cuando se ocultan el Calendario y la Barra de Iconos).

```
Case 991 To 1185: 'Número Interfaz para Reportes
                  Interfaz = 6
                  PIconos.Picture = ImageIcons.ListImages(6).Picture
                  PIconos.ToolTipText = "Reportes"
                  IFondoC.Height = 1695
                  IFondoC.Width = 3650
                  IFondoC.Left = 0
                  IFondoC.Top = 0
                  MainForm.Height = 1650
                  Lmsg.Caption = "Seleccione un reporte."
                  '************************************Muestra
                  FrGraficos.Visible = True
                  IFondoC.Visible = True
                  '************************************Oculta
                  ImOnOff.Visible = False
                  FrBotones.Visible = False
                  FrCitas.Visible = False
                  FrCitas2.Visible = False
                  FrAgenda.Visible = False
                  FrContactos.Visible = False
                  FrContactos2.Visible = False
                  FrConfig.Visible = False
                  FrNotas.Visible = False
                  FrNotas2.Visible = False
                  OBPendientes.Visible = False
                  OBTerminados.Visible = False
```

Cuando se selecciona la interfaz de Configuración, el método es el mismo, solo que en este caso se muestra el *Frame* que contiene los controles de Configuración, que aunque son más que los botones de los Reportes, también se encuentran agrupados en un solo *Frame*.

```
Case 1186 To 1350: 'Número Interfaz para configuración
                   Interfaz = 7
                   ImOnOff.Visible = False
                   PIconos.Picture = ImageIcons.ListImages(7).Picture
                   PIconos.ToolTipText = "Configuración"
                   IFondoC.Height = 1695
                   IFondoC.Width = 3650
                   IFondoC.Left = 0
                   IFondoC.Top = 0
                   MainForm.Height = 1650
                   Lmsg.Caption = "Modifique la configuración deseada y de click en guardar."

                   '************************************Muestra
                   FrConfig.Visible = True
                   IFondoC.Visible = True
                   '************************************Oculta
                   FrBotones.Visible = False
                   FrGraficos.Visible = False
                   FrCitas.Visible = False
                   FrCitas2.Visible = False
                   FrAgenda.Visible = False
                   FrContactos.Visible = False
                   FrContactos2.Visible = False
                   'LVContactos.Visible = False
                   FrNotas.Visible = False
                   FrNotas2.Visible = False
                   OBPendientes.Visible = False
                   OBTerminados.Visible = False

                   'LVNotas.Visible = False
                   'LVCitas.Visible = False
```

Algo interesante sobre la barra de iconos que cabe resaltar, es que como se mencionó previamente, se trata de una imagen que contiene todos los iconos de las interfaces, entonces cuando el usuario pasa el cursor sobre ella, el *ToolTipText* es único para todos los iconos, es decir, es necesario cambiar el texto del mismo, según la ubicación del cursor.
Utilizando el mismo principio que el de cambio de interfaces, estos textos se cambian de acuerdo a la coordenada "*Y"* del Evento *MouseMove*, con ayuda de un *Select Case*.

```
Private Sub PIconos_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
 Select Case Y
   Case 0 To 180: PIconos.ToolTipText = "Reloj/Calendario"
   Case 181 To 370: PIconos.ToolTipText = "Agendas"
   Case 370 To 570: PIconos.ToolTipText = "Contactos"
   Case 571 To 795: PIconos.ToolTipText = "Citas"
   Case 796 To 990: PIconos.ToolTipText = "Notas"
   Case 991 To 1185: PIconos.ToolTipText = "Reportes"
   Case 1186 To 1350: PIconos.ToolTipText = "Configuración"
 End Select

End Sub
```

Continuando con las funciones de la ventana principal, algo que ayudó a un diseño minimalista fue que ésta tuviera la opción de ocultar el calendario, ya que de lo contrario, ocuparía mucho espacio en la pantalla, opacando el efecto.
Ocultar el calendario es realmente sencillo, solo es necesario modificar la propiedad *Height* de la ventana principal, a un tamaño que ya no permita visualizarlo. Y para ayudar al usuario, un botón (etiqueta) con un triángulo simulando una flecha que apunta hacia arriba se agregó en la interfaz principal.
Cuando el usuario hace clic en él, su forma cambia a un triángulo apuntando hacia abajo para dar una idea más intuitiva de que algo se desplegará cuando se dé clic en él.
Ahora bien, una vez que se oculta el Calendario, la marquesina aún queda visible, al igual que la barra de iconos. El diseño se ve minimalista ya, pero aún falta un paso para serlo. Dejar el reloj visible únicamente es más estético, entonces el clic en la “flecha” que oculta esta información funcionará en dos pasos. En el primero se ocultará únicamente el Calendario y en el segundo se ocultarán tanto la marquesina como la barra de iconos. Y al tercer clic, cuando la flecha ya apunta hacia abajo nuevamente, toda la interfaz se desplegará completa para mostrar Calendario, marquesina y barra de iconos.
Este ciclo se maneja a través de condicionales, revisando el texto de la etiqueta y el tamaño de la ventana. También fue necesario cambiar el *ToolTipText* en cada ocasión para facilitar el entendimiento de su funcionamiento.

```
Public Sub LDisplay_Click()
 If LDisplay.Caption = "6" Then  'Con marquesina
  If MainForm.Height = 1430 Then
   MainForm.Height = 1650
   MainForm.Width = 3915
   LDisplay.ToolTipText = "Mostrar Calendario"
  Else
   LDisplay.Caption = "5"   'Todo
   LDisplayS.Caption = "5"
   MainForm.Height = 3675
   LDisplay.ToolTipText = "Ocultar Todo"
  End If
 Else
  LDisplay.Caption = "6"   'Nada
  LDisplayS.Caption = "6"
  MainForm.Height = 1430
  MainForm.Width = 3600
  LDisplay.ToolTipText = "Mostrar Menú / Marquesina"
 End If

End Sub
```

En las interfaces que incluyen la función de búsqueda, el evento que dispara la función es el *Change* de cada caja de texto que es sensible de búsqueda. La función Buscar es descrita posteriormente.

Timer1

Este *timer* es el encargado de llevar el conteo de la hora mostrada en el Reloj. Como las etiquetas que se utilizan para el mismo están separadas en unidades y decenas, es necesario realizar una verificación de la hora, minutos y segundos para añadir un “0” cuando dichas mediciones den resultados menores a 10. Se realiza lo mismo para el caso del día del mes.
Este *timer* inicialmente tiene un intervalo de 30, y se utiliza para hacer el efecto de conteo rápido inicial. Todos los valores son colocados en cero, se calcula la hora actual con minutos y segundos, incluyendo el día del mes.
A continuación se incrementan todos los elementos en uno cada vez (en cada intervalo del *timer*) hasta que cada uno llega al valor actual a mostrar donde se detiene.
Una vez “ajustados” todos los elementos, la propiedad “intervalo” del *timer* es colocada en 1000, que corresponde a un segundo estándar.
Este efecto es sencillo pero visualmente atractivo.

```
Private Sub Timer1_Timer()
 Dim Hora1, Min1, Sec1, Dia1
 Hora1 = Hour(Now)
 Min1 = Minute(Now)
 Sec1 = Second(Now)
 Dia1 = Day(Date)

 If Min1 = 0 Or Min1 = 30 Then Call GuardaEst

 If cont < 59 Then
  If cont <= Val(Left(Hora1, 1)) Then
   If Hora1 > 9 Then
     MainForm.LHG1.Caption = cont
     MainForm.LHN1.Caption = cont
   Else
     MainForm.LHG1.Caption = "0"
     MainForm.LHN1.Caption = "0"
   End If
  End If
  If cont <= Val(Right(Hora1, 1)) Then
   MainForm.LHG2.Caption = cont
   MainForm.LHN2.Caption = cont
  End If
  If cont <= Val(Left(Min1, 1)) Then
   If Min1 > 9 Then
    MainForm.LMG1.Caption = cont
    MainForm.LMN1.Caption = cont
   Else
    MainForm.LMG1.Caption = "0"
    MainForm.LMN1.Caption = "0"
   End If
  End If
  If cont <= Val(Right(Min1, 1)) Then
   MainForm.LMG2.Caption = cont
   MainForm.LMN2.Caption = cont
  End If
  If cont <= Val(Sec1) Then
```

Timer2

Este *timer* se emplea para desplazar el texto en la marquesina. Tiene un valor en la propiedad de intervalo de 30, y lo que hace es ir disminuyendo la propiedad *Left* de la etiqueta de la marquesina en 15 unidades cada vez. Cuando la suma del ancho total de la etiqueta más la posición *Left* son menores o iguales a cero, entonces la propiedad *Left* es enviada nuevamente al final de la ventana. Esta suma en la comparación debe hacerse pues la marquesina cambia constantemente su tamaño dependiendo de los eventos y las interfaces seleccionadas.

```
Private Sub Timer2_Timer()
 If Lmsg.Left + Lmsg.Width > 0 Then
  Lmsg.Left = Lmsg.Left - 15
 Else
  Lmsg.Left = 3950
 End If
End Sub
```

TimerAgente

Un tercer *timer*, llamado TimerAgente, es el encargado de medir en segundo plano, el tiempo de inactividad del usuario de acuerdo al valor establecido en el campo correspondiente de la interfaz de configuración.

Esto se detecta al verificar el estado del botón *Play*. Si el botón tiene el triángulo de Play, quiere decir que en ese momento no se está midiendo ningún tema y el *Timer* se activa para comenzar la medición. Esto se hace desde el código del botón *Play*.

Cuando el botón cambia a Pausa, entonces este *timer* es desactivado y por tanto reiniciado a cero en el conteo que compara los minutos sin actividad con el valor de la configuración.

Pero si el *timer* continúa su conteo y llega al valor de la configuración, entonces se muestra una ventana indicando al usuario cuántos minutos han transcurrido sin realizar alguna actividad y se hacen sugerencias de los temas que no han sido trabajados últimamente.

Cuando la ventana de aviso se muestra, se coloca el número de minutos que han transcurrido sin actividad. Sin embargo, al inicio no importaba si acababa de suceder o tenía dos horas de haber aparecido, el mensaje seguía mostrando lo mismo si no se cerraba la ventana y se mostraba de nuevo. Entonces la información mostrada sobre la inactividad era obviamente errónea.

Esto representó un problema interesante de resolver pues había que detectar primeramente si dicha ventana estaba activa en la pantalla o no, para poder mostrar información sobre el tiempo de inactividad.

Afortunadamente, para el Agente existe una rutina que permite analizar las ventanas que se encuentran actualmente abiertas y permite revisar sus títulos, con lo cual es posible saber si la ventana del mensaje está abierta y puede modificarse el texto en cualquier momento para actualizar el tiempo de inactividad.

En este *timer* también se verifica el status de los recordatorios de citas. Lo primero que se hace es verificar la fecha de cada cita en el listado y si coincide con la actual entonces el siguiente paso es revisar al hora de la cita.

```
Private Sub TimerAgente_Timer()
 Dim X, i, cad, res, TiempoVencido, Texto, j
 Dim LVGen As ListView
 ContMin = ContMin + 1
    With FEstAct.LVResumen.ListItems(1)
     .SubItems(9) = Val(FEstAct.LVResumen.ListItems(1).SubItems(9)) + 1
    End With
  FRecordatorioCita.LVRecordatorios.ListItems.Clear
  For j = 1 To 2
   If j = 1 Then Set LVGen = LVCitas Else Set LVGen = LVCitasP

  For i = 1 To LVGen.ListItems.Count
  TiempoVencido = DateDiff("n", Now, LVGen.ListItems(i).SubItems(3) & " " & LVGen.ListItems(:
   If Date = LVGen.ListItems(i).SubItems(3) And InStr(1, LVGen.ListItems(i).Text, "*") = 0 Tl
    If Val(LVGen.ListItems(i).SubItems(5)) > 0 Then
     LVGen.ListItems(i).SubItems(5) = Val(LVGen.ListItems(i).SubItems(5)) - 1
     GoTo siguiente
    End If
    If TiempoVencido < 0 Then
     Texto = "Hace " & -1 * TiempoVencido \ 60 & " hrs. " & -1 * Round(TiempoVencido Mod 60,
    Else
     Texto = "Cita en " & TiempoVencido \ 60 & " hrs. " & Round(TiempoVencido Mod 60, 2) & "
    End If

     LVGen.ListItems(i).Selected = True
     With FRecordatorioCita.LVRecordatorios.ListItems.Add(, , Texto)
      .SubItems(1) = LVGen.ListItems(i).Text
      .SubItems(2) = LVGen.ListItems(i).SubItems(1)
      .SubItems(3) = LVGen.ListItems(i).SubItems(4)
      .SubItems(4) = LVGen.ListItems(i).SubItems(3)
      .SubItems(5) = LVGen.ListItems(i).SubItems(2)
      .SubItems(6) = LVGen.Name & "-" & i
     End With
```

El tiempo para mostrar los recordatorios también se configura en la interfaz de Configuración y en este *timer* lo que se hace es comparar (*DateDiff*) la hora actual con la hora de cada una de las citas en el listado de Citas menos el tiempo configurado. Si la diferencia es menor de cero, entonces la cita está vencida y si es mayor o igual a cero, entonces está a tiempo o justo a tiempo.
No importa cuál sea el caso, si la cita está programada para ese día, entonces será agregada en un listado de la ventana que mostrará las citas vencidas y por vencer, donde se permite aplazar y omitir citas.

```
 FRecordatorioCita.LInfo.Caption = "Las siguientes citas han vencido o están por vencer en
 FRecordatorioCita.CBPosp.ListIndex = 0
 If FRecordatorioCita.LVRecordatorios.ListItems.Count <> 0 Then FRecordatorioCita.LVRecorda

 Call m_UtilVentanas.EnumTopWindows(LVVentanas)
 X = Module1.IsFormLoaded(MensajeInactividad)

If X <> 0 Then
   If ContMin = 1 Then
    MensajeInactividad.LMje.Caption = "Ha transcurrido " & ContMin & " minuto sin que se a
   ElseIf ContMin >= 60 Then
    MensajeInactividad.LMje.Caption = "Han transcurrido " & ContMin \ 60 & " hora(s) " & C
   Else
    MensajeInactividad.LMje.Caption = "Han transcurrido " & ContMin & " minutos sin que se
   End If

End If

 If ContMin Mod Val(TBStandBy.Text) = 0 Then
  If BPlayAg.Caption = "4" Then
   MensajeInactividad.Height = 1680
   If ContMin = 1 Then
    MensajeInactividad.LMje.Caption = "Ha transcurrido " & ContMin & " minuto sin que se a
   ElseIf ContMin >= 60 Then
    MensajeInactividad.LMje.Caption = "Han transcurrido " & ContMin \ 60 & " hora(s) " & C
   Else
    MensajeInactividad.LMje.Caption = "Han transcurrido " & ContMin & " minutos sin que se
   End If
```

Esta ventana, consta de una *ListView*, tres botones y un *ComboBox*. El *ListView* contiene la lista de citas del día actual o incluso de días anteriores pero que no han sido terminadas u omitidas.
El botón Omitir lo que hace es marcar la cita seleccionada de tal manera que cuando el *timer* verifica las citas para ese día no la muestre en el recordatorio.
Esto se hace colocando un “*” en el título de la cita. Así, cuando se ejecuta la revisión en cada intervalo del *timer*, si la cita tiene esa marca se ignora y si es lo contrario se agrega al listado a mostrar.
Otro botón es el de “cerrar”, qué únicamente oculta esta ventana, hasta que el *timer* procesa nuevamente las citas (al minuto) y aparecen nuevamente las que vencen ese día.
El último botón está ligado con el *ComboBox*, este botón se utiliza para posponer las citas un tiempo determinado, según se seleccione en el *ComboBox* (5,10,15,30 minutos, 1,2,4 horas).
Para que el aplazamiento funcione, es necesario agregar un campo en el *ListView* de la cita que permanece oculto al usuario pero que es vital para su funcionamiento. Lo que se hace al otorgar un aplazamiento, es colocar en ese campo el número en minutos (transformado de horas si es necesario) del tiempo seleccionado por el usuario para posponer la cita. Este número será disminuido en uno cada minuto que pase y el *timer* analice las citas. Si al revisar este campo en una cita, tiene un valor de cero, entonces el tiempo ha vencido y es hora de mostrarla en el recordatorio. En caso contrario, únicamente se hará la resta de uno y continuará su conteo.
Así con cada cita en el listado.

```
Private Sub BPosponer_Click()
Dim i, numInd, cad, MinPos
 For i = LVRecordatorios.ListItems.Count To 1 Step -1
  If LVRecordatorios.ListItems(i).Selected = True Then
    cad = LVRecordatorios.ListItems(i).SubItems(6)
    numInd = Val(Right(cad, Len(cad) - InStr(1, cad, "-")))
    Select Case CBPosp.ListIndex
     Case 0: MinPos = 5
     Case 1: MinPos = 10
     Case 2: MinPos = 15
     Case 3: MinPos = 30
     Case 4: MinPos = 60
     Case 5: MinPos = 120
     Case 6: MinPos = 240
    End Select
   If InStr(1, LVRecordatorios.ListItems(i).SubItems(6), "LVCitasP") <> 0 Then
    MainForm.LVCitasP.ListItems(numInd).SubItems(5) = MinPos
   Else
    MainForm.LVCitas.ListItems(numInd).SubItems(5) = MinPos
   End If
   LVRecordatorios.ListItems.Remove (i)
  End If
 Next i
If LVRecordatorios.ListItems.Count = 0 Then Unload Me
End Sub
```

Hasta aquí con la descripción de las funciones principales de la ventana principal que es la más compleja.

A continuación se describirán ventanas adicionales que apoyan y facilitan el uso del programa.
Cuando después de un tiempo sin actividad el *Timer* del Agente dispara el mensaje que informa al usuario los minutos que han transcurrido sin activar tareas, lo hace a través de una ventana llamada MensajeInactividad. Esta ventana tiene la apariencia de un simple mensaje informativo, con dos botones que ayudan a responder la pregunta que se muestra en una etiqueta. “Han transcurrido ‘x’ minutos sin que se active ningún tema. ¿Desea que le recomiende alguna actividad?” Si el usuario responde que No, entonces la ventana se cierra y se mostrará posteriormente cuando el tiempo del recordatorio llegue nuevamente a cero.
Si el usuario responde que sí, entonces el tamaño de la ventana cambia y se despliegan dos *ListViews* que contienen información sobre las actividades que han tenido poco o nulo movimiento los últimos días.

Para determinar de una manera más sencilla el movimiento en los temas de la Agenda, existe un campo al final de cada tema que contiene el número de veces que el tema es activado. Entonces cuando un tema se selecciona, se activa el cronómetro y se detiene, además de aumentar el número de minutos trabajados del mismo, este último campo es incrementado en uno.
De esta manera, podemos saber con certeza y rapidez los temas que estando pendientes, han sido los menos trabajados. Entonces se hace un filtro y todos los temas que tengan un número de activaciones de uno, son agregados a la lista de “Temas que no han sido revisados últimamente”.

```
For i = 1 To MainForm.LVTemasL.ListItems.Count
 If MainForm.LVTemasL.ListItems(i).SubItems(9) < 2 Then
  With LVTemasSinR.ListItems.Add(, , MainForm.LVTemasL.ListItems(i).SubItems(5))
   .SubItems(1) = MainForm.LVTemasL.ListItems(i).SubItems(1)
   .SubItems(2) = MainForm.LVTemasL.ListItems(i).SubItems(4)
   .SubItems(3) = "Laboral"
  End With
 End If
 If DateDiff("d", MainForm.LVTemasL.ListItems(i).SubItems(5), Now) > 15 Then
  With LVTemas15d.ListItems.Add(, , MainForm.LVTemasL.ListItems(i).SubItems(5))
   .SubItems(1) = MainForm.LVTemasL.ListItems(i).SubItems(1)
   .SubItems(2) = MainForm.LVTemasL.ListItems(i).SubItems(4)
   .SubItems(3) = "Laboral"
  End With
 End If
Next i
For i = 1 To MainForm.LVTemasP.ListItems.Count
 If MainForm.LVTemasP.ListItems(i).SubItems(9) < 2 Then
  With LVTemasSinR.ListItems.Add(, , MainForm.LVTemasP.ListItems(i).SubItems(5))
   .SubItems(1) = MainForm.LVTemasP.ListItems(i).SubItems(1)
   .SubItems(2) = MainForm.LVTemasP.ListItems(i).SubItems(4)
   .SubItems(3) = "Personal"
  End With
 End If
 If DateDiff("d", MainForm.LVTemasP.ListItems(i).SubItems(5), Now) > 15 Then
  With LVTemas15d.ListItems.Add(, , MainForm.LVTemasP.ListItems(i).SubItems(5))
   .SubItems(1) = MainForm.LVTemasP.ListItems(i).SubItems(1)
   .SubItems(2) = MainForm.LVTemasP.ListItems(i).SubItems(4)
   .SubItems(3) = "Personal"
  End With
 End If
Next i
LsinR.Caption = " Temas que no han sido revisados últimamente: (" & LVTemasSinR.ListItems.Co
L15d.Caption = " Temas con más de 15 días como pendientes: (" & LVTemas15d.ListItems.Count &
If LVTemas15d.ListItems.Count = 0 Then MsgBox "No se encontraron temas pendientes con más de
If LVTemasSinR.ListItems.Count = 0 Then MsgBox "No se encontraron temas pendientes con menos
```

Al Terminar el tema, este campo es reiniciado a cero, para el caso de que pudiera ser activado nuevamente, pueda realizarse el seguimiento posteriormente.

El otro listado mostrado contiene los temas que continúan pendientes y tienen más de 15 días desde su creación.
Estos dos criterios se obtuvieron de una encuesta rápida entre los usuarios del programa, pues los plazos más comunes en el área son de 15 días y aquellos temas que no se activan más de una vez probablemente se estén olvidando pero no pueden ni deben perderse de vista.

Como extra, se hace un cálculo rápido entre los minutos que se ha permanecido sin activar un tema desde que se inició el programa y los minutos que en teoría debieran trabajarse diariamente para tener un estimado del porcentaje de la jornada laboral que no se ha activado algún tema.
Este porcentaje es simplemente informativo y depende totalmente del usuario la confiabilidad del mismo. Es decir, si el usuario trabaja pero no activa ningún tema para llevar el control de su tiempo, al final del día el sistema podría decirle que permaneció inactivo el 100% de su tiempo, lo cual no es verdadero.

La información que se da en esta ventana es una de las herramientas de apoyo para la mejor administración personal del tiempo del usuario. Las otras son la ventana de Estadística de Uso y la ventana de Estadística de Actividades.

Estadística de Uso.

Esta ventana muestra al usuario información acerca del uso de la PC, que es básicamente el resultado del monitoreo del Agente durante el uso del programa desde que se inicia.
Esta información contiene la fecha y hora de inicio del programa, el tiempo total de uso, el programa más usado con su tiempo en minutos y el programa menos usado también con su tiempo en minutos, un listado de las ventanas que se abren durante el día y otro con los programas abiertos en el mismo tiempo y finalmente un total de Ventanas abiertas, activas y programas abiertos en el día.

Toda esta información puede ayudar al usuario a conocer qué programa es el que más utiliza y las ventanas a las que dedica más tiempo, para que él mismo determine si es necesario modificar sus prioridades y dedicar más tiempo a otras actividades o continuar así. Muchas veces podemos sorprendernos al revisar estos datos pues al trabajar por inercia damos por hecho que distribuimos bien nuestro tiempo en las actividades diarias, pero la verdad es que dedicamos más tiempo a actividades que no ayudan a nuestro desarrollo y/o desempeño profesional y que sin embargo son necesarias para una buena salud mental. La clave está en detectarlo y asignarle un tiempo más adecuado y equilibrado con respecto al resto de actividades.
Estos datos, además de ser mostrados en los listados, pueden visualizarse en unas sencillas gráficas de barras que facilitan su comprensión y uso. A continuación la descripción de las funciones principales de estas dos ventanas de Estadística de uso.

Como se mencionó en la función *Load* de la ventana principal, existe una variable global llamada HoraInicio, que cuando el programa se carga almacena la hora. Esta información es útil en esta ventana de Estadística de Uso y es lo primero que se carga en su campo de texto TInicio al cargar la ventana. Mediante la función *DateDiff* se hace una resta de minutos entre la Hora de Inicio almacenada y la hora actual para determinar el número de minutos que el programa ha estado en uso y el resultado se coloca en su campo de texto Tuso después de ser convertido a horas y minutos mediante la misma función *DateDiff*.

```
Private Sub Form_Load()
 Dim i, max, Tiempo, a
 TiempoUso = DateDiff("n", MainForm.HoraInicio, Now)
 TInicio.Text = MainForm.HoraInicio
 If DateDiff("n", MainForm.HoraInicio, Now) > 59 Then
  TUso.Text = DateDiff("n", MainForm.HoraInicio, Now) \ 60 & " h " & DateDiff("n", Ma
 Else
  TUso.Text = "0 h " & DateDiff("n", MainForm.HoraInicio, Now) & " m"
 End If

 With ListView1.ColumnHeaders.Add(, , "Ventanas Abiertas", 6000)
  .Tag = "Programas"
 End With
 With ListView1.ColumnHeaders.Add(, , "Tiempo (min)", 1200)
  .Tag = "Tiempo"
 End With
 With ListView1.ColumnHeaders.Add(, , "Clase", 0)
  .Tag = "Clase"
 End With

 ListView1.HideColumnHeaders = False
 ListView1.GridLines = True


 For i = 1 To MainForm.LVVentanas.ListItems.Count
  Tiempo = MainForm.LVVentanas.ListItems(i).SubItems(1)
  With ListView1.ListItems.Add(1, , MainForm.LVVentanas.ListItems(i))
   If Len(Tiempo) = 1 Then
    .SubItems(1) = "000" & MainForm.LVVentanas.ListItems(i).SubItems(1)
   ElseIf Len(Tiempo) = 2 Then
    .SubItems(1) = "00" & MainForm.LVVentanas.ListItems(i).SubItems(1)
   ElseIf Len(Tiempo) = 3 Then
    .SubItems(1) = "0" & MainForm.LVVentanas.ListItems(i).SubItems(1)
   End If
    .SubItems(2) = MainForm.LVVentanas.ListItems(i).SubItems(2)
  End With
```

Posteriormente mediante un ciclo se toman los datos del *ListView* LVVentanas de la ventana principal y se ordenan de manera descendente respecto al tiempo empleado por cada ventana.
Como se recordará de la función *Load* en la ventana principal, al finalizar se hace un llamado a una función EnumTopWindows con este objeto. La función se describe en la sección Clase, pero para adelantar un poco, lo que se guarda en el objeto con esta función son los títulos de las ventanas que se abren, así como los tiempos y tipos o clases de las mismas.

Después de ordenar se hace un llamado al procedimiento “SeparaClases”.
Este procedimiento tiene dos objetivos, el primero es clasificar las ventanas abiertas de acuerdo al programa y el segundo es llenar el listado de programas abiertos con dicha información.
Clasificar las ventanas es un proceso relativamente sencillo gracias a que previamente fue guardada la clase o tipo de ventana en el *ListView* Ventanas, cada vez que era agregada una ventana, mediante la función EnumTopWindows. Con este dato únicamente es necesario utilizar un *Select Case* y clasificar todas las posibles ventanas en solo 24 programas, que son los más representativos y más utilizados en el área de aplicación del programa:
SAP, Explorador de Windows®, Internet Explorer®, Mozilla Firefox, Microsoft Outlook®, Microsoft Exce®l, Microsoft Word®, Microsoft PowerPoint®, Adobe Acrobat®, Microsoft Wordpad®, Java®, Bloc de Notas®, Microsoft Photo Editor®, Microsoft Paint®, WinZip®, Calculadora, Aplicación de Sistema, Símbolo de Sistema (MS-DOS®), Quicktime®, ZG_View, Administrador de Tareas de Windows®, Microsoft Visual Basic®, Saturnus y Otros.
Las clases de dichos programas, entre otras, son: SAP_FRONTEND_SESSION, ExploreWClass, IEFrame, MozillaWindowsClass, rctrl_renwnd32, MS-SDIa, XLMAIN, OpusApp, PP11FrameClass, AcrobatSDIWindow, WordPadClass, SunAwtFrame, Notepad, MSPhotoEditor32MainClass,

MSPaintApp, WinZipWClass, SciCalc, SysFader, ConsoleWindowClass, QuickTimePlayerMain, ZG-View 4.21 Instance Class, #32770, wndclass_desked_gsk, ThunderRT6FormDC.

```
Private Sub SeparaClases()
Dim i, j, cad, Existe, TipoV
LVHistorial.ListItems.Clear

For i = 1 To ListView1.ListItems.Count
 cad = ListView1.ListItems(i).SubItems(2)

  Select Case cad
   Case "SAP_FRONTEND_SESSION": TipoV = "SAP"
   Case "wndclass_desked_gsk": TipoV = "Microsoft Visual Basic"
   Case "ThunderRT6FormDC": TipoV = "iRobot"
   Case "ExploreWClass": TipoV = "Explorador de Windows"
   Case "IEFrame": TipoV = "Internet Explorer"
   Case "Notepad": TipoV = "Bloc de Notas"
   Case "#32770": TipoV = "Administrador de Tareas de Windows"
   Case "rctrl_renwnd32": TipoV = "Microsoft Outlook"
   Case "SunAwtFrame": TipoV = "Java, AVON"
   Case "MS-SDIa", "XLMAIN": TipoV = "Microsoft Excel"
   Case "MozillaWindowClass": TipoV = "Mozilla Firefox"
   Case "OpusApp", "bosa_sdm_Microsoft Office Word 11.0": TipoV = "Microsoft Word"
   Case "SciCalc": TipoV = "Calculadora"
   Case "AcrobatSDIWindow", "Ghost": TipoV = "Adobe Acrobat"
   Case "MSPhotoEditor32MainClass": TipoV = "Microsoft Photo Editor"
   Case "MSPaintApp": TipoV = "Microsoft Paint"
   Case "WinZipWClass": TipoV = "Winzip"
   Case "SysFader", "DV2ControlHost": TipoV = "Aplicación de Sistema"
   Case "WordPadClass": TipoV = "Microsoft Wordpad"
   Case "ConsoleWindowClass": TipoV = "Símbolo de Sistema (MS-DOS)"
   Case "PP11FrameClass": TipoV = "Microsoft Powerpoint"
   Case "QuickTimePlayerMain": TipoV = "Quicktime"
   Case "ZG-View 4.21 Instance Class": TipoV = "ZG-View"
   Case Else: TipoV = "Otros"
  End Select
 Existe = 0
```

En otros obviamente se clasificarán todas aquellas ventanas cuya clase no se encuentra en alguna de las anteriores. El nombre del programa es guardado en una variable llamada TipoV, y será utilizada más adelante.

Antes de colocar el programa al que pertenece la ventana en el listado, se verifica si ya existe en el mismo, en caso positivo, se incrementa en uno el número de instancias del programa, en caso negativo se agrega al listado y se coloca un 1 como número de instancias.

Con esto, solo resta calcular el tiempo de uso de cada programa, lo cual se determina con un promedio, dividiendo el total de la suma del tiempo empleado en cada ventana del programa y dividiéndolo entre el número de instancias del mismo. Esto proporciona un número si bien no exacto, muy cercano a la realidad, pues de lo contrario las cantidades se dispararían y no serían correctas.

Al resultado se le hacen ajustes de formato para mostrarlo solo con dos decimales y los datos se ordenan en el *ListView*.

Al estar ordenados por tiempo, entonces como paso final resta mostrar los programas y ventanas más y menos usados, lo cual se obtiene fácilmente tomando el primer y el último elemento de cada *ListView*, y colocando la información (nombre y tiempo) en los campos de texto correspondientes.

El número de programas y ventanas abiertas se obtiene del número de elementos de cada *ListView*.

```
 For j = 1 To LVHistorial.ListItems.Count
  If TipoV = LVHistorial.ListItems(j) Then Existe = j: Exit For
 Next j
 If Existe = 0 Then
  With LVHistorial.ListItems.Add(, , TipoV)
   .SubItems(1) = Val(ListView1.ListItems(i).SubItems(1))
   .SubItems(2) = 1
  End With
 Else
  With LVHistorial.ListItems(j)
   .SubItems(1) = Val(.SubItems(1)) + Val(ListView1.ListItems(i).SubItems(1))
   .SubItems(2) = Val(.SubItems(2)) + 1
  End With
 End If
Next i

For i = 1 To LVHistorial.ListItems.Count
  With LVHistorial.ListItems(i)
   .SubItems(1) = Trim(Str(Round(Val(LVHistorial.ListItems(i).SubItems(1)) / Val(LVHist
   If InStr(1, .SubItems(1), ".") = 0 Then .SubItems(1) = .SubItems(1) + ".00"
   If Len(Right(.SubItems(1), Len(.SubItems(1)) - InStr(1, .SubItems(1), "."))) = 1 The
   If Len(Left(.SubItems(1), InStr(1, .SubItems(1), ".") - 1)) = 0 Then
    .SubItems(1) = "0000" & LVHistorial.ListItems(i).SubItems(1)
   ElseIf Len(Left(.SubItems(1), InStr(1, .SubItems(1), ".") - 1)) = 1 Then
    .SubItems(1) = "000" & LVHistorial.ListItems(i).SubItems(1)
   ElseIf Len(Left(.SubItems(1), InStr(1, .SubItems(1), ".") - 1)) = 2 Then
    .SubItems(1) = "00" & LVHistorial.ListItems(i).SubItems(1)
```

Existe un botón en la ventana que permite actualizar la información mostrada, pues conforme pasan los minutos estos datos se tienen que reajustar para ser más precisos.
El código del botón ejecuta las mismas instrucciones que cuando la ventana es cargada.

Otro botón en la ventana llamado “Ver Gráfico” abre un formulario que muestra una gráfica de barras simple con los datos explicados en los párrafos anteriores. Una gráfica de barras por lo general consta de un eje “X” y un eje “Y”, y unos rectángulos cuyas dimensiones coinciden con los valores a graficar en una escala previamente definida.
Dado que por cuestiones ya explicadas al inicio, se debía evitar en lo posible el uso de controles adicionales que requirieran instalación especial, se optó por la opción de realizar manualmente el proceso de graficado. *Visual Basic*® ya contiene en sus herramientas de creación de formularios el objeto *Shape*, en cuyas propiedades se encuentra *Style*, que ya incluye entre las opciones de figuras prediseñadas un rectángulo, cuyo tamaño se modifica cambiando las propiedades *Height* y *Width*.
Para el área de graficado, se colocó un *Frame* con color de fondo negro sobre el cual se crearon diez rectángulos de diferentes colores (en gradiente siendo el rojo el más importante) en los bordes y sin relleno. Es decir solo se muestran los contornos coloreados de los mismos y otras tantas diez etiquetas que contendrán el valor máximo de la barra que le corresponde.

Los datos que se pueden graficar son únicamente los siguientes:

- Los 10 programas más usados
- Los 10 programas menos usados
- Las 10 ventanas más usadas
- Las 10 ventanas menos usadas

Y las opciones a mostrar son porcentajes y minutos.

Por cuestiones de espacio y mejor visualización, las barras se colocaron horizontalmente, dejando el eje “X”, el del porcentaje o los minutos, en el extremo superior y el eje “Y”, el de las 10 series, en el extremo izquierdo. Los ejes solo son objetos línea y tienen dimensiones y ubicación fijas.
Al igual que las etiquetas que marcan el 0, el 50 y el 100 %.

Para seleccionar las opciones de graficado, las 4 opciones mencionadas arriba se despliegan en un *ComboBox*, y la opción de Porcentaje/Minutos en un *OptionButton*, para restringir al usuario la selección a uno solo. Ambos objetos se encuentran encerrados en un *Frame*.

El código del graficado es relativamente sencillo, pues solo debe hacerse la conversión de la medida de la serie a la propiedad W*idth* de la barra que le corresponde, tomando el máximo como 100 en el caso del porcentaje y el máximo de minutos de la caja de texto "TiempoUso" de la ventana Estadística de uso.

Para facilitar el graficado se creó una función llamada "Grafica" y que recibe como parámetros 3 datos: Programa, Porcentaje y MasMenos.
Programa determina si se graficarán los datos de los programas o de las ventanas abiertos.
Porcentaje determina si se graficarán los porcentajes o los minutos y
MasMenos determina si se toman los 10 valores de mayor a menor (10 más) o de menor a mayor (10 menos).

```
Public Sub Grafica(Programa As Integer, Porcentaje As Integer, MasMenos)
 Dim i, ListaG As ListView, TopN, C, Inicio, Fin, Paso
 If Programa = 1 Then
  Set ListaG = FEstUso.LVHistorial
 Else
  Set ListaG = FEstUso.ListView1
 End If

 If ListaG.ListItems.Count < 10 Then TopN = ListaG.ListItems.Count Else TopN = 10
 For i = 0 To 9
  Bars(i).Width = 0
  Etiquetas(i).Caption = ""
 Next i

If MasMenos = 1 Then
  Inicio = 1
  Fin = TopN
  Paso = 1
Else
  Inicio = ListaG.ListItems.Count
  Fin = 1
  Paso = -1
End If
 j = Inicio
 For i = 1 To TopN
   If FEstUso.TiempoUso > 0 Then
    C = (Val(ListaG.ListItems(j).SubItems(1)) * Factor) / FEstUso.TiempoUso
   Else
    C = 0
   End If
   If Porcentaje = 1 Then
    Bars(i - 1).Width = (C * 11295) / 100
    Etiquetas(i - 1).Caption = ListaG.ListItems(j).Text & "  - (" & Round(C, 2) & " :
```

Al revisar el parámetro Programa, se elige el *ListView* de la ventana Estadística de Uso correcto.
Después, dado que podríamos tener menos de 10 ventanas o programas abiertos, se revisa el número de elementos del *ListView*, si fueran menos de 10, entonces en la variable de TopN se coloca este número, de lo contrario se coloca 10.
Se limpian las etiquetas y las barras se inician en cero.
A continuación se lee el parámetro MasMenos y con esto las variables Inicio, Fin y Paso que se utilizan en el ciclo que leerá los datos del *ListView* quedan establecidas, si es de mayor a menor

(valor 1) tomamos como inicio el 1, fin será la variable TopN calculada previamente y el paso será 1, es decir, el ciclo incrementará en uno cada vez.
Si la gráfica es de menor a mayor (valor 0) entonces el inicio será el total de elementos del *ListView* (en este caso no es TopN pues si hay más de 10 elementos en el *ListView*, graficaríamos los mismos datos que si fuera de mayor a menor), el fin sería 1 y el paso -1, es decir, el ciclo disminuirá en uno cada vez.
A continuación, en el ciclo se calculan los valores a graficar, multiplicando el valor que tiene en el *ListView* por un factor (variable global) entre el tiempo de uso total. El resultado se multiplica por el tamaño total del *Width* de la barra y dividiendo entre 100 para el caso de poner porcentajes y se deja así para el caso de los minutos.
Este resultado a la propiedad *Width* del rectángulo (barra) correspondiente.
Y esto se hace con cada una de las 10 barras.

Esta función “Grafica”, es llamada cada vez que el usuario selecciona un tipo de Gráfico del ComboBox, enviando según sea el caso los parámetros adecuados.
Por ejemplo, cuando el usuario selecciona la opción:

Los 10 programas más usados y la opción de Porcentaje está seleccionada, los parámetros que se envían a la función son:
Programa: 1, Porcentaje: 1 y MasMenos: 1. Además la variable global factor se establece en 100.
Si fueran las 10 ventanas menos usadas y en Minutos, los parámetros serían:
Programa: 2, Porcentaje: 2 y MasMenos: 0. Además la variable global factor se establece en 11295.

```
Private Sub CBOpProgVent_Click()
 Select Case CBOpProgVent.ListIndex
  Case 0, 1:
            If OBPorcentaje.Value = True Then
             TituloGraf = "Porcentaje del Tiempo Total que los programas han estado abiertos."
             Factor = 100
             If CBOpProgVent.ListIndex = 0 Then Grafica 1, 1, 1 Else Grafica 1, 1, 0
            Else
             TituloGraf = "Tiempo total en minutos que los programas han estado abiertos."
             Factor = 11295
             If CBOpProgVent.ListIndex = 0 Then Grafica 1, 2, 1 Else Grafica 1, 2, 0
            End If

  Case 2, 3:
            If OBPorcentaje.Value = True Then
             TituloGraf = "Porcentaje del Tiempo Total que las ventanas han estado abiertas."
             Factor = 100
             If CBOpProgVent.ListIndex = 2 Then Grafica 2, 1, 1 Else Grafica 2, 1, 0
            Else
             TituloGraf = "Tiempo total en minutos que las ventanas han estado abiertas."
             Factor = 11295
             If CBOpProgVent.ListIndex = 2 Then Grafica 2, 2, 1 Else Grafica 2, 2, 0
            End If
```

Al seleccionar una gráfica diferente, también la etiqueta del título del gráfico se modifica.
Cuando el usuario abre esta ventana del Gráfico por primera vez, lo que se muestra por default es la gráfica de los 10 programas más usados, para lo cual se establece el Factor en 100 y se llama a la función “Grafica” con los parámetros 1,1,1. Además se selecciona el primer elemento del *ComboBox* (índice 0) que corresponde a esta gráfica.

Estadística de Actividades

La estadística o Reportes de Actividades utiliza las agendas para mostrar al usuario un resumen de las actividades que ha realizado durante el uso del programa. Este resumen se crea a partir de un registro diario de las actividades y calculando los promedios según los días evaluados.
La interfaz consta de un *ListView* que muestra dicho registro diario y que almacena por día un registro o línea y tres Frames que engloban y resumen esta información: Promedios de Actividades Laborales, Actividades Personales y Totales.
En el caso de los primeros, cada *Frame* a su vez contiene los promedios de las Actividades Pendientes, Trabajadas, Terminadas y el Tiempo promedio empleado en dichas actividades.
Para los Totales, se muestran el total de Actividades Pendientes, Trabajadas, Terminadas, el Tiempo Trabajado, el número de Días Evaluados y el Tiempo Inactivo.
Este último es importante en el sentido de que informa al usuario el promedio de tiempo que ha permanecido sin realizar o de manera más correcta, sin registrar, alguna actividad.
Dado que se trata de promedios, la información será más certera conforme más se utilice y más días sean evaluados.
Los datos se muestran en *Textboxes* y adicional al *ListView* y los *Frames*, existen dos botones en la esquina inferior derecha, uno para mostrar la gráfica de la información que aquí se muestra y otro para cerrar esta ventana.

El código de esta ventana es muy simple y no requiere mucha explicación.
Tenemos únicamente tres funciones:
Cerrar, que en realidad lo que hace es ocultar esta ventana en lugar de cerrarla. Esto lo hace para poder colocar información nueva en el *ListView* mientras se ejecuta el programa, ya que si se descarga de la memoria, entonces cuando las funciones de la Agenda intenten agregar información en esta ventana sería necesario abrirla cada vez y ocultarla. De esta manera, agilizamos la visualización de la información sin comprometer la velocidad o el uso de memoria del programa.
La función que se ejecuta al dar clic en el botón de Gráfico de Actividades lo único que hace es mostrar el formulario o ventana correspondiente, en este caso llamado FGraficoActividades.
Y finalmente la función *Activate* del formulario mismo, que se disparará cada vez que el usuario de clic en el botón Estadística de Actividades de la interfaz Reportes, ya descrita previamente.
Esta función es la que realiza el cálculo de todos los promedios mostrados.
Lo primero que es necesario revisar y almacenar, es el número total de registros del *ListView*, o en otras palabras el número de días que se están evaluando.
A continuación, mediante un sencillo ciclo *For*…*Next* se suman en nueve variables los valores que se muestran en el *ListView* por día, es decir se calculan los totales de cada clasificación: Actividades Laborales Pendientes, Trabajadas, Terminadas y Tiempo Laboral Trabajado, Actividades Personales Pendientes, Trabajadas, Terminadas y Tiempo Personal Trabajado, y por último Tiempo Inactivo por día.
Ahora solo es necesario colocar esta información en las *Textboxes* correspondientes para los promedios, no sin antes realizar la división entre el total que se calculó previamente y que a su vez será colocado en la *Textbox* de días evaluados.
Para los totales de Actividades Pendientes, Trabajadas, Terminadas y el Tiempo Trabajado del tercer *Frame*, se suman los promedios totales correspondientes de las Agendas Laboral y Personal.
El promedio de Tiempo Inactivo se calcula igualmente dividiendo el total entre el número de días evaluados y mostrando el resultado con un redondeo de dos decimales.

```
Private Sub BCerrar_Click()
 Me.Hide
End Sub

Private Sub BGraficoAct_Click()
 FGraficoActividades.Show
End Sub

Private Sub Form_Activate()
Dim i, ActTrab, ActTer, TiemTrab, TiemInac, Total
Dim ActLabPen, ActLabTrab, ActLabTer, TiemLabTrab, ActPerPen, ActPerTrab, ActPerTer, TiemPerTrab
Total = LVResumen.ListItems.Count
 For i = 1 To Total
  ActLabPen = Val(LVResumen.ListItems(i).SubItems(1)) + ActLabPen
  ActLabTrab = Val(LVResumen.ListItems(i).SubItems(2)) + ActLabTrab
  ActLabTer = Val(LVResumen.ListItems(i).SubItems(3)) + ActLabTer
  TiemLabTrab = Val(LVResumen.ListItems(i).SubItems(4)) + TiemLabTrab
  ActPerPen = Val(LVResumen.ListItems(i).SubItems(5)) + ActPerPen
  ActPerTrab = Val(LVResumen.ListItems(i).SubItems(6)) + ActPerTrab
  ActPerTer = Val(LVResumen.ListItems(i).SubItems(7)) + ActPerTer
  TiemPerTrab = Val(LVResumen.ListItems(i).SubItems(8)) + TiemPerTrab
  TiemInac = Val(LVResumen.ListItems(i).SubItems(9)) + TiemInac
 Next i
 TbALPe.Text = Round((ActLabPen / Total), 1)
 TbALTr.Text = Round((ActLabTrab / Total), 1)
 TbALTe.Text = Round((ActLabTer / Total), 1)
 TbALTi.Text = Round((TiemLabTrab / Total), 2)

 TbAPPe.Text = Round((ActPerPen / Total), 1)
 TbAPTr.Text = Round((ActPerTrab / Total), 1)
 TbAPTe.Text = Round((ActPerTer / Total), 1)
 TbAPTi.Text = Round((TiemPerTrab / Total), 2)

 TBActPend.Text = (Val(TbALPe.Text) + Val(TbAPPe.Text))
 TBActTrab.Text = (Val(TbALTr.Text) + Val(TbAPTr.Text))
 TBActTerm.Text = (Val(TbALTe.Text) + Val(TbAPTe.Text))
```

El Gráfico de las Actividades es un tanto distinto al de la Gráfica de Uso, ya que en este caso, los datos a graficar son menos dinámicos y complicados.
Para empezar, la gráfica se encuentra desplegada con barras verticales y no horizontales, pues como máximo tendremos cinco columnas mostrando información.
El eje de las “X”, contendrá las Actividades a graficar y el eje de la “Y” el porcentaje o el número de las mismas.
El diseño es similar a la Gráfica de uso, con un fondo negro y mostrando unos rectángulos cuya altura se modifica para representar las barras. Aquí los rectángulos tienen activa las propiedades *FillColor* y *FillStyle* que los rellena con algún color y diseño específico. Elegí un rayado en diagonal alternado en cada columna para mostrar un poco de color en la gráfica sin parecer tampoco un simple rectángulo de color.

Al igual que la gráfica de uso, un *ComboBox* le permitirá al usuario elegir la información que graficará, aunque en este caso las opciones son Total de Actividades, Actividades Laborales, Actividades Personales y Tiempos. Así mismo se cuenta con dos *OptionButton* que permiten elegir Mostrar Porcentajes o Mostrar Cantidades. Y finalmente el infalible botón “Cerrar”.
El resto de elementos de la Gráfica son las etiquetas que mostrarán los títulos de las barras, los ejes y el título de la gráfica.
Primeramente, al cargar el formulario, se selecciona el índice 0 del *ComboBox* que hará que se muestre en la gráfica el Total de Actividades, llamando automáticamente a la función *Click* del *ComboBox*. Esta función elegirá, mediante un *Select Case* el índice del combo seleccionado, pudiendo ser cuatro opciones, mencionadas previamente:

Total de Actividades.
Para esta gráfica las columnas o barras empleadas serán cinco, y sus títulos serán Laborales, Personales, Pendientes, Trabajadas y Terminadas.

Después se evalúa con un condicional si se trata de porcentaje o cantidades y se manda a llamar a la función GraficaAct para cada columna con los parámetros correspondientes que se explicarán más adelante al describir la función.

Actividades Laborales
De la misma manera, para esta opción se establecen los títulos de las barras y que en esta opción serán solo tres las graficadas: Pendientes, Trabajadas y Terminadas. Se determina si se trata de porcentaje o cantidad y se llama a la función GraficaAct para cada una de ellas.

Actividades Personales
Funciona de manera idéntica a la anterior, con excepción de que los parámetros que se envían a la función GraficaAct serán los correspondientes al de las Actividades Personales.

Tiempos
Para el caso de los tiempos, las barras a graficar serán nuevamente tres, con los textos Personales, Laborales e Inactivo.
Para cada columna se llama a la función GraficaAct y el parámetro “Total” que se envía, será en este caso el valor 540, que corresponde al número de minutos que tiene una jornada laboral promedio de 9 horas.

```
Private Sub CBOpActividades_Click()
Dim TiempoTot
 Select Case CBOpActividades.ListIndex
  Case 0:   'Actividades Totales
             Etiquetas(0).Caption = "Laborales"
             Etiquetas(1).Caption = "Personales"
             Etiquetas(2).Caption = "Pendientes"
             Etiquetas(3).Caption = "Trabajadas"
             Etiquetas(4).Caption = "Terminadas"

            If OBPorcentaje.Value = True Then
             TituloGraf = "Actividades en Total (Porcentaje)"
             GraficaAct Val(FEstAct.TbALPe.Text) + Val(FEstAct.TbALTe.Text), 100, 0, 1, Val(FEst.
             GraficaAct Val(FEstAct.TbAPPe.Text) + Val(FEstAct.TbAPTe.Text), 100, 1, 1, Val(FEst.
             GraficaAct Val(FEstAct.TBActPend.Text), 100, 2, 1, Val(FEstAct.TBActPend.Text) + Va.
             GraficaAct Val(FEstAct.TBActTrab.Text), 100, 3, 1, Val(FEstAct.TBActPend.Text) + Va.
             GraficaAct Val(FEstAct.TBActTerm.Text), 100, 4, 1, Val(FEstAct.TBActPend.Text) + Va.
            Else
             TituloGraf = "Actividades en Total (Cantidad)"
             GraficaAct Val(FEstAct.TbALPe.Text) + Val(FEstAct.TbALTe.Text), 6015, 0, 2, Val(FEst
             GraficaAct Val(FEstAct.TbAPPe.Text) + Val(FEstAct.TbAPTe.Text), 6015, 1, 2, Val(FEst
             GraficaAct Val(FEstAct.TBActPend.Text), 6015, 2, 2, Val(FEstAct.TBActPend.Text) + Va
             GraficaAct Val(FEstAct.TBActTrab.Text), 6015, 3, 2, Val(FEstAct.TBActPend.Text) + Va
             GraficaAct Val(FEstAct.TBActTerm.Text), 6015, 4, 2, Val(FEstAct.TBActPend.Text) + Va

            End If
  Case 1:   'Actividades Laborales
             Etiquetas(0).Caption = "Pendientes"
             Etiquetas(1).Caption = "Trabajadas"
             Etiquetas(2).Caption = "Terminadas"
             Etiquetas(3).Caption = ""
             Etiquetas(4).Caption = ""

            If OBPorcentaje.Value = True Then
             TituloGraf = "Actividades Laborales (Porcentaje)"
```

La función GraficaAct se creó para graficar más fácilmente cada columna o barra de la Gráfica y funciona de la siguiente manera.
Tiene cinco parámetros:

ValorInd: Que contiene esencialmente el valor a graficar de la columna.
Factor: Que corresponde al valor máximo de altura que puede tener la barra. 100 para porcentajes y 6015 para cantidades.

NumBarra: el número de barra o columna a graficar, y que corresponde al índice de la matriz de los cinco rectángulos que se usan para graficar.
Porcentaje: Que se usa como bandera para saber si se grafica porcentaje (1) o cantidades (2)
Total: Que contiene el valor máximo de la columna, según lo calculado.

Debido a la manera en la que funcionan los objetos *Shape* en la interfaz, al momento de cambiar la altura del rectángulo, el eje "X" que se toma se encuentra en la parte superior del formulario y por tanto, la base de aquél se encuentra en la parte superior. En otras palabras, si se manejara únicamente la altura del rectángulo sin corrección, entonces la gráfica estaría de cabeza.
Para esto fue necesario hacer una resta al final del cálculo en la función.

Esto también obligó a 'resetear' las barras al inicio de la función, colocando el valor inicial de su propiedad *Top* en 480 que corresponde a la línea del límite superior de la gráfica.

```
Public Sub GraficaAct(ValorInd, Factor, NumBarra, Porcentaje, Total)
 Dim i, C

 Bars(NumBarra).Top = 480

        If Porcentaje = 1 Then
           L100.Caption = "100%"
           L50.Caption = "50%"
           L0.Caption = "0%"
           L100.Tag = Total
        ElseIf Porcentaje = 2 Then
           L100.Caption = Total
           L50.Caption = Round(Total / 2, 2)
           L0.Caption = "0"
           L100.Tag = Total
        End If
        If Total = 0 Then
         C = 0
        Else
         C = (ValorInd * Factor) / Val(L100.Tag)
        End If
        If Porcentaje = 1 Then
            Bars(NumBarra).Height = (C * 6015) / 100
            Etiquetas(NumBarra).Caption = Etiquetas(NumBarra).Caption & "  - (" & Round(C, 2) & '
        Else
            Bars(NumBarra).Height = C
            If Porcentaje <> 3 Then Etiquetas(NumBarra).Caption = Etiquetas(NumBarra).Caption & '
        End If
        Bars(NumBarra).Top = Bars(NumBarra).Top + (6015 - Bars(NumBarra).Height)
End Sub
```

Después, las etiquetas que muestran los valores del eje "Y" son rellenadas con los valores de porcentajes o cantidades según corresponda el parámetro Porcentaje.

Si por algún motivo el valor del parámetro Total es cero, entonces debe realizarse directamente la asignación de cero a la variable C (que se utiliza para calcular la altura de la barra) pues de lo contrario la fórmula realizaría una división por cero, causando, como es bien conocido un error fatal.
Ahora, para calcular la altura de la barra, solamente se utiliza una regla de tres con los parámetros que conocemos, multiplicando el ValorInd por el Factor y dividiendo entre el Total. Y para el caso de porcentajes en lugar de valores, se necesita realizar una operación adicional que consiste en multiplicar por 6015 el valor de c y dividirlo entre 100.
Finalmente para realizar la corrección mencionada al inicio, se realiza una resta de 6015 menos el valor de la altura de la barra y esto se suma a la propiedad *Top* de la misma, lo cual en realidad está desplazando el rectángulo a la parte inferior de la ventana a una distancia que permita colocar la parte "superior" del rectángulo (la altura) en el eje "X".

Cuando se utilizan solo tres columnas para graficar, las otras dos deben graficarse en cero, lo cual es enviado en el parámetro ValorInd.

Finalmente, el botón cerrar, descarga el formulario de la memoria con la función *Unload*.

Ventana Agregar Evento

Para agregar un evento en el calendario, se explicó previamente que el usuario puede dar doble clic en cualquier día del calendario mostrado y esto le abrirá una ventana que después de seleccionar mes, día, evento y repetición, lo agregará al listado de eventos de la ventana Configuración de Eventos antes de guardarlo.

La ventana muestra en un *Frame* las opciones a elegir de mes y día mediante dos *Comboboxes*, un *Textbox* permite agregar los datos del evento y un *Checkbox* le da al usuario la opción de elegir si el evento se repetirá anualmente o no.
En la parte inferior se encuentran dos botones, Agregar y Cancelar.
El botón Agregar abrirá la ventana de Configuración ya con los datos cargados del evento, y el botón Cancelar cerrará esta ventana sin agregar un solo dato en el archivo de eventos.

Al presionar Agregar, se verifica que el campo de texto del Evento no esté vacío y que el día esté también seleccionado, de lo contrario sale de la función y no agrega nada.

Es importante que la información de los eventos se encuentre correctamente ordenada cronológicamente, para lo cual se utilizo un pequeño truco. Si creamos una cadena que contenga primeramente el número del mes y posteriormente concatenamos el día y convertimos el resultado en un número, tendremos una manera rápida de ordenar fechas por simple comparación.
Por ejemplo, si queremos agregar un evento que se llevará a cabo el 16 de junio, el número que nos resulta de realizar la cadena como se mencionó sería 616. Cualquier número generado por una fecha anterior dará como resultado un número menor, por ejemplo 16 de enero es 116. Al igual que cualquier fecha posterior generará un número mayor 17 de diciembre sería 1217.

Entonces, una vez determinado que el mes y el día están seleccionados en la ventana, se almacena en una variable (MEv) el valor que estos producen. Después se buscará en el listado uno a uno los eventos creando en otra variable (ML) el número que crean las fechas y al encontrar el primer valor que sea mayor que la fecha del evento que queremos agregar, se detiene la búsqueda y almacenamos el índice donde se encontró en la variable índice, que fue inicializado antes del ciclo en -1.
Si al terminar el ciclo índice sigue siendo -1, entonces no se encontró fecha mayor y el evento debe colocarse al final de la lista, por lo tanto el lugar que le corresponde será el número de elementos de la lista. De lo contrario, el evento será agregado en el lugar que le corresponda según la variable índice.
Para formar la cadena del evento a agregar, concatenamos las primeras tres letras del mes en mayúsculas, un guión, el día, otro guión y el texto del evento.

```
Private Sub BAgregar_Click()
Dim ML, MEv, Indice
    CadEv = TBEvento.Text
    If CadEv = "" Then Exit Sub
    If CBDia.ListIndex = -1 Then Exit Sub
    MEv = Val(Str(CBMes.ListIndex + 1) & CBDia.List(CBDia.ListIndex))

    Indice = -1
    If FConfig.LEventos.ListCount <> 0 Then
     For i = 0 To FConfig.LEventos.ListCount - 1
      Select Case Left(FConfig.LEventos.List(i), 3)
       Case "ENE": ML = 1
       Case "FEB": ML = 2
       Case "MAR": ML = 3
       Case "ABR": ML = 4
       Case "MAY": ML = 5
       Case "JUN": ML = 6
       Case "JUL": ML = 7
       Case "AGO": ML = 8
       Case "SEP": ML = 9
       Case "OCT": ML = 10
       Case "NOV": ML = 11
       Case "DIC": ML = 12
      End Select
      ML = ML & Mid(FConfig.LEventos.List(i), 5, 2)

      If Val(MEv) < Val(ML) Then
       Indice = i
       Exit For
      End If
     Next i
    End If
    If Indice = -1 Then Indice = FConfig.LEventos.ListCount
    If CBRepAnual.Value = 1 Then CadEv = CadEv & "*"  ' se repite anualmente
    FConfig.LEventos.AddItem UCase(Left(CBMes.List(CBMes.ListIndex), 3)) & "-" & LTrim(CBDia.List
    FConfig.LEventos.ListIndex = Indice
          ""
```

Se mencionó también que en la ventana existe un *Checkbox* que permite elegir si el evento se repetirá anualmente. Esto fue pensado principalmente en los cumpleaños, ya que son eventos que normalmente el usuario quiere tener en su calendario para no olvidarlo, pero que al repetirse anualmente sería tedioso estar capturando año con año todos los cumpleaños que queremos recordar. Así que una manera sencilla para el usuario de hacer esto fue colocarle el *Checkbox* que lo que hace en esta parte del código es colocar un asterisco al final del nombre del evento. Cuando se explique el procedimiento para guardar la lista de eventos en un archivo se explicará el uso de este asterisco adicional.
Una vez agregado el evento en el *Listbox* de la ventana FConfig, en el lugar correspondiente, se le asigna el índice a la propiedad *ListIndex* del mismo, haciendo que cuando la ventana se muestre este evento esté seleccionado.
Se limpian las variables públicas mes y día, y en la variable pública “Cambio” de FConfig se le asigna el valor *True*. Esta variable le indica a la ventana que se ha realizado un cambio en el listado de manera que cuando se quiera cerrar la ventana, haya un cambio y no se haya guardado el archivo, se le pregunte al usuario si desea guardar los cambios o no.

También se tuvo que pensar en la validación de los días que aparecen en el listado correspondiente en esta ventana, de manera que, cuando el usuario elige un mes, entonces de acuerdo al mes elegido se asigna a la variable Fin el valor máximo de días del mes.
Posteriormente mediante un ciclo, estos días son agregados al *ComboBox* de días después de haberlo limpiado.
Así mismo, al momento de agregar los días, fue importante colocar un cero antes del dígito del día para los días menores a 10. Esto se hace con la finalidad de que la cadena generada para el ordenamiento tenga la información correcta, porque de lo contrario la comparación podría ser errónea, por ejemplo. Si la fecha que queremos agregar es el 4 de julio y no colocamos el cero antes del cuatro, entonces el valor generado sería 74, y si lo comparamos con un 15 de mayo que tendría un valor de 515, nos diría que el 15 de mayo es mayor, cuando en realidad debe ser menor.

Esto se corrige fácilmente al agregar el cero previo dándonos un valor correcto de 704, que es visiblemente mayor que 515. Otra opción pudiera haber sido transformar las fechas en números julianos, pero esta opción es más práctica para los fines empleados.

```
Private Sub CBMes_Click()
Dim Fin
CBDia.Clear
 Select Case CBMes.ListIndex
  Case 0, 2, 4, 6, 7, 9, 11: Fin = 31
  Case 3, 5, 8, 10: Fin = 30
  Case 1: Fin = 28
 End Select

 For i = 1 To Fin
  If i < 10 Then
   CBDia.AddItem "0" & i
  Else
   CBDia.AddItem i
  End If
 Next i
 CBDia.Enabled = True


End Sub
```

Cuando toda esta información ha sido validada y manipulada correctamente, ya puede ser mostrada la ventana FConfig.
Esta ventana, le muestra al usuario los eventos para administrarlos, es decir, agregar, eliminar, editar y guardar eventos.
Esta misma ventana se muestra en la interfaz de configuración, pero en esta ocasión es llamada por la ventana de AgregarEvento pues fue llamada desde el calendario.

La interfaz consta de un *ListView* que muestra el listado de eventos para ese año, 4 etiquetas a manera de botón (Agregar, Eliminar, Editar, Guardar), y un botón de Cerrar.
En la parte inferior está una etiqueta que mostrará al usuario la ruta del archivo de eventos usado actualmente. Esto solo es de carácter informativo.
El funcionamiento de esta ventana es como sigue:
Cuando la ventana es llamada desde AgregarEvento por el calendario, los datos se encuentran ya en el listado y lo único que resta es guardar para actualizar el archivo de eventos. Sin embargo, si esta ventana es llamada desde la configuración, es necesario manejarlo de manera distinta, pues ahora esta ventana será la que llame a AgregarEvento cuando el usuario haga clic en el botón Agregar.
Cuando se carga esta ventana, lo primero que hace es buscar si el archivo al que hace referencia la variable global “NombreEventos” del *MainForm* existe mediante la función *Dir*.
Si el archivo existe, entonces los datos son cargados uno a uno en el *Listbox* LEventos de la ventana.

```
    If FConfig.LEventos.ListCount <> 0 Then
     For i = 0 To FConfig.LEventos.ListCount - 1
      Select Case Left(FConfig.LEventos.List(i), 3)
       Case "ENE": ML = 1
       Case "FEB": ML = 2
       Case "MAR": ML = 3
       Case "ABR": ML = 4
       Case "MAY": ML = 5
       Case "JUN": ML = 6
       Case "JUL": ML = 7
       Case "AGO": ML = 8
       Case "SEP": ML = 9
       Case "OCT": ML = 10
       Case "NOV": ML = 11
       Case "DIC": ML = 12
      End Select
      ML = ML & Mid(FConfig.LEventos.List(i), 5, 2)

      If Val(MEv) < Val(ML) Then
       Indice = i
       Exit For
      End If
     Next i
    End If
    If Indice = -1 Then Indice = FConfig.LEventos.ListCount
    FConfig.LEventos.AddItem v0, Indice
  Loop
 Close #1
```

Para hacer más fácil el manejo de los eventos que se repiten anualmente, un archivo adicional es creado con su ruta almacenada en la variable global “NombreEventosRep”. Esto será explicado cuando se describa el botón Guardar de esta interfaz.
Entonces, con la misma función *Dir*, se determina si el archivo de eventos repetidos existe o no.
Si existe, se abre el archivo y uno a uno se leen los eventos repetitivos, y para colocarlos en el listado ya cargado con los eventos del año, se utiliza el mismo procedimiento descrito previamente con las cadenas de mes y día y haciendo las comparaciones. De esta manera el listado final que se muestra al usuario, contendrá los eventos del año más los eventos repetitivos anualmente correctamente ordenados y únicamente identificables por el asterisco al final del nombre del evento.

Botón Agregar.

El botón (etiqueta) Agregar básicamente muestra la ventana AgregarEvento, pero para hacerlo, primeramente es necesario establecer los valores de Mes y Día para que el usuario solo tenga que escribir el nombre del evento. Sin embargo, para hacerlo es necesario saber si el llamado de Agregar se hizo desde la propia ventana configuración de Eventos o fue llamado desde el doble clic de un día en la interfaz de calendario. La diferencia entre estas dos opciones radica en que si se hizo el llamado mediante doble clic en el día, entonces este día y mes son los que deben ser colocados en los *Comboboxes* correspondientes de la ventana de AgregarEvento. Por el contrario, si se hizo desde esta misma ventana, el día y el mes que deben ser mostrados son los actuales.
Esto es simple cuando se coloca una variable llamada DesdeCal, que al hacer doble clic en un día del calendario mostrado, es establecida en verdadero. Y por lo tanto, en el código del botón Agregar que estamos analizando, mediante un condicional verificamos el valor de la variable

DesdeCal y si es verdadero colocamos en el día y mes los valores de las variables “día” y “mes” que fueron llenados al hacer doble clic.
En caso contrario, obtenemos los valores mediante la función *Month*(*Date*) para obtener el mes actual y *Day*(*Date*) para el día.
Para que el *Combobox* muestre el valor correcto, debemos restar la unidad pues los índices en los combos comienzan en cero.
Finalmente cambiamos el color de la etiqueta/botón Agregar a un verde intenso, para indicar al usuario que es el botón que se encuentra activo en ese momento y mostramos la ventana AgregarEvento con el método *Show*.

```
Public Sub LAdd_Click()
If DesdeCal = True Then
  Cambio = True
  AgregarEvento.CBMes.ListIndex = mes - 1
  AgregarEvento.CBDia.ListIndex = dia - 1
ElseIf FConfig.LEventos.ListIndex = -1 Then
 AgregarEvento.CBMes.ListIndex = Month(Date) - 1
 AgregarEvento.CBDia.ListIndex = Day(Date) - 1
End If
LAdd.ForeColor = &HFF00&
AgregarEvento.Show

End Sub
```

Botón Eliminar

Eliminar un evento es sencillo, el usuario debe elegir el evento que quiere eliminar de la lista, y entonces al presionar el botón, se le pide que confirme la eliminación y si su respuesta es positiva, simplemente se utiliza el método *RemoveItem* del *ListBox* con el número de índice del evento seleccionado y establecemos la variable “Cambio” a verdadero, para recordar al usuario que debe guardar el archivo antes de cerrar la ventana.
Al igual que con el botón Agregar, se modifica el color de la etiqueta para indicar que el botón se encuentra actualmente activo.

```
Private Sub LDelete_Click()
Dim res
 LDelete.ForeColor = &HFF00&
 res = MsgBox("¿Está seguro que desea eliminar este evento?", vbOKCancel + vbQuestion, "Eliminar"
 If res = vbOK Then
  LEventos.RemoveItem LEventos.ListIndex
  Cambio = True
 End If
 LDelete.ForeColor = &HC0C0C0
End Sub
```

Botón Editar.

La edición de un evento se realiza de la siguiente manera.
En primer lugar el usuario debe elegir el evento en el listado, y después de cambiar el color de la etiqueta como en los casos anteriores, verificamos el índice del evento seleccionado.
En caso de que el índice sea -1, es decir que no se encuentre ningún evento seleccionado, entonces se le notifica al usuario mediante un mensaje y sale de la función.
Si no es así, entonces tomamos el elemento de la lista, y lo dividimos en 3 secciones (que están separadas por un guión), la primera contiene el mes, la segunda el día y la tercera el evento. A partir de aquí tenemos los elementos necesarios para mostrar la ventana Agregar Evento que ahora dirá Editar Evento.
Con los datos del día y del mes activamos el índice de los *Comboboxes* correspondientes del formulario AgregarEvento, y en el *Textbox* colocamos el valor de cadena del evento.

Si esta cadena tiene un asterisco al final, entonces la *Checkbox* de Repetir Anualmente tiene que activarse.
Listo, ahora solo resta que el usuario haga los cambios correspondientes en esta información y cuando de *click* en Agregar entonces el evento original será removido del listado, hemos guardado previamente en una variable el índice en el *ListBox* del elemento seleccionado y el evento modificado se agrega en el lugar correcto como si se tratase de un evento nuevo.

```
Private Sub LEdit_Click()
Dim CadEv
 LEdit.ForeColor = &HFF00&
 If LEventos.ListIndex = -1 Then LEventos.ListIndex = LEventos.ListCount - 1
 CadEv = InputBox("Edite el evento como quiera que aparezca:", "Editar Evento", LEventos.List(LEv
 If CadEv <> "" Then LEventos.List(LEventos.ListIndex) = CadEv
 Cambio = True
 LEdit.ForeColor = &HC0C0C0
End Sub
```

Botón Guardar

La información que se encuentra en el *ListBox*, Lista de Eventos, son los que serán almacenados en un archivo de texto, mejor dicho, en dos archivos de texto.
En un archivo se guardarán todos los eventos que no sean repetitivos y en el otro únicamente los que se repitan anualmente, o en otras palabras, los que tengan un asterisco al final de la cadena del evento.
Básicamente el proceso es el siguiente, se abren los dos archivos de texto para escritura con la instrucción *Open…For Output* como #1 y #2. Después se leen uno a uno los elementos del *ListBox* y si al final se encuentra un asterisco, se almacena el elemento en #2, si no en #1, en ambos casos se utiliza la instrucción *Write*.
Al finalizar el ciclo, el botón de guardar se regresa a su color original (que se había cambiado a verde al dar *click* en él) y se manda llamar la función CreaCal teniendo como parámetros los valores de las variables globales MesCalAct, AñoCalAct y día 1. Esto mostrará nuevamente el calendario que se encontraba antes de agregar el evento, pero con la diferencia de que ahora contendrá el evento que se acaba de crear en caso de que coincida con el mes del evento.
Originalmente tenía que cerrarse y abrirse la aplicación cada vez para mostrar los cambios, pero como una mejora al programa, al añadir la función CreaCal, esto ya no es más necesario.

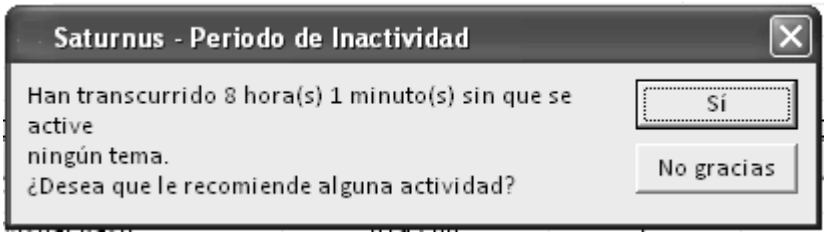
```
Public Sub LSave_Click()
 LSave.ForeColor = &HFF00&
 Open NombreEventos For Output As #1
 Open NombreEventosRep For Output As #2
  For i = 0 To LEventos.ListCount - 1
   If Right(LEventos.List(i), 1) = "*" Then
    Write #2, LEventos.List(i)
   Else
    Write #1, LEventos.List(i)
   End If
  Next i
 Close #1
 Close #2
 MsgBox "Archivo guardado.", vbInformation + vbOKOnly
 Cambio = False
 LSave.ForeColor = &HC0C0C0
 MainForm.Lmsg = ""
 MainForm.DomRES = CalculaSemanaSanta(AñoCalAct)
 CreaCal 1, MesCalAct, AñoCalAct, MainForm.DomRES

End Sub
```

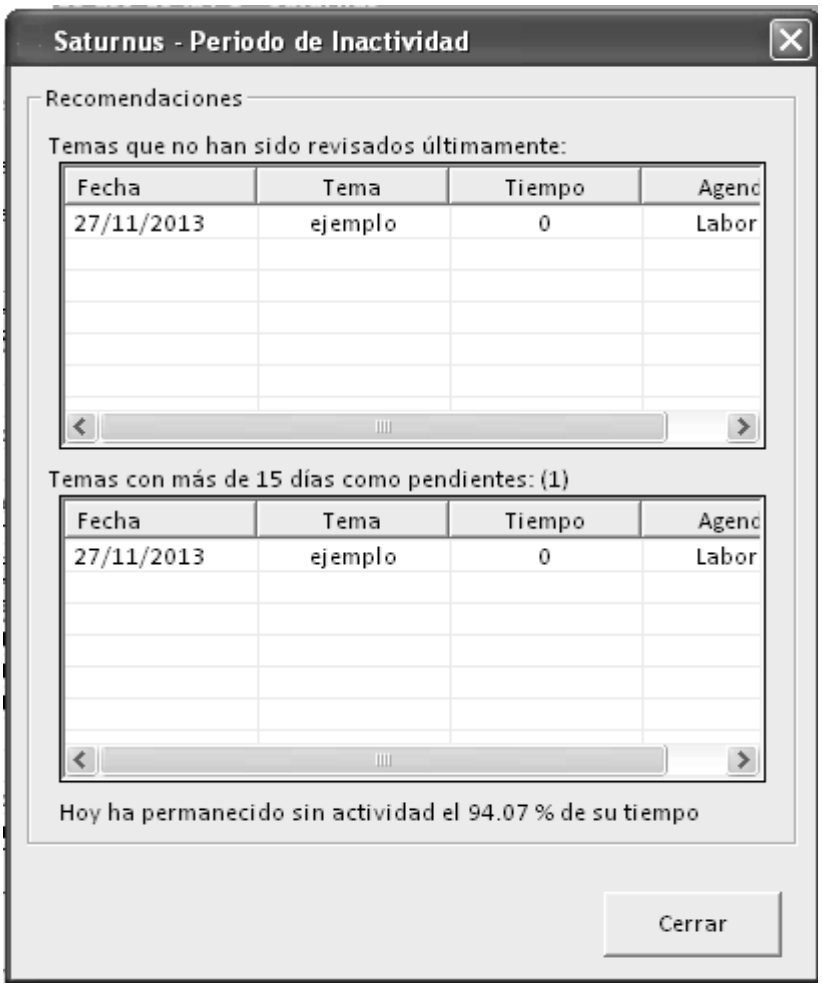
Finalmente el botón Cerrar simplemente descarga el formulario con la instrucción *Unload*, no sin antes verificar el estado de la variable “Cambio”. En el caso de que su valor sea verdadero, quiere decir que se han introducido cambios en el listado y no ha sido guardado aún, por lo que se le pregunta al usuario mediante un *MsgBox* con respuesta si desea guardar los cambios. En caso afirmativo llamamos a la función Guardar (como si se diera *click* en el botón guardar).

Hay una última ventana que aún no ha sido descrita, y se trata de aquella que informa al usuario que no ha realizado ninguna tarea en un tiempo determinado y que se encuentra registrado en la ventana Configuración.
El cómo y cuando se activa esta ventana ya fue explicado previamente cuando se habló del TimerAgente, y ahora se explicará como funciona internamente este formulario.

Cuando se muestra el formulario al vencer el tiempo de inactividad, éste contiene únicamente una etiqueta y dos botones. La etiqueta le informa al usuario el tiempo que ha estado inactivo, y los dos botones son el de “Sí” y el de “No Gracias”. El de “No Gracias” no requiere mayor explicación, pero el de “Sí”, se utiliza para responder afirmativamente la pregunta que se le hace al usuario de si desea que el programa le haga alguna sugerencia para emplear su tiempo.



Lo primero que se hace es redimensionar el formulario, pues al inicio tiene la apariencia de un mensaje informativo cualquiera, pero al dar *click* en Aceptar, la ventana “crece” para mostrar los dos *ListViews* y las etiquetas que se describieron anteriormente.
Así mismo es necesario “subir” el formulario en la pantalla del usuario pues ahora es más alto y podría no mostrarse completo al redimensionarlo. Esto se hace muy fácilmente al modificar la propiedad *Top* del mismo. En este caso únicamente le restamos 2000 unidades al valor que tenía.



Ahora lo único que resta es revisar los listados de temas laborales y temas personales de los *ListViews* correspondientes de la ventana principal, que como se recordará, ya contienen una columna que almacena el número de veces que un tema ha sido revisado. Si este número es menor a dos, es decir, si ha sido revisado una sola vez, entonces este elemento es agregado al *ListView* de “Temas que no han sido revisados últimamente”.
Y si al realizar la resta de la fecha actual con la fecha del elemento en turno nos da un valor mayor a 15 (días) entonces este elemento es agregado al *ListView* de “Temas con más de 15 días como pendientes”.
Obviamente si algún tema cumple con ambas condiciones, éste será agregado en ambos *ListViews*.
En las etiquetas de los títulos de los *ListViews* se agrega el número de elementos encontrados por *ListView* y en la etiqueta inferior se mostrará el tiempo de inactividad en porcentaje mediante un cálculo rápido de regla de tres con el valor del tiempo de inactividad del día (almacenado en la columna 9 del primer elemento del *ListView* LVResumen de la ventana FEstAct y el valor 540 que corresponde a las 9 horas laborales diarias que tenemos como promedio en el área de aplicación del programa.

Y un truco final, para ocultar la etiqueta y los botones iniciales que informan al usuario del tiempo que lleva sin actividad al redimensionar el formulario, simplemente se modifica el valor *Top* del *Frame* que contiene los *ListViews*, etiquetas y botones que muestran el resumen de temas sugeridos.

```
Private Sub OKButton_Click()
 Dim i
 LVTemasSinR.ListItems.Clear
 LVTemas15d.ListItems.Clear
 Me.Top = Me.Top - 2000

 For i = 1 To MainForm.LVTemasL.ListItems.Count
  If MainForm.LVTemasL.ListItems(i).SubItems(9) < 2 Then
   With LVTemasSinR.ListItems.Add(, , MainForm.LVTemasL.ListItems(i).SubItems(5))
    .SubItems(1) = MainForm.LVTemasL.ListItems(i).SubItems(1)
    .SubItems(2) = MainForm.LVTemasL.ListItems(i).SubItems(4)
    .SubItems(3) = "Laboral"
   End With
  End If
  If DateDiff("d", MainForm.LVTemasL.ListItems(i).SubItems(5), Now) > 15 Then
   With LVTemas15d.ListItems.Add(, , MainForm.LVTemasL.ListItems(i).SubItems(5))
    .SubItems(1) = MainForm.LVTemasL.ListItems(i).SubItems(1)
    .SubItems(2) = MainForm.LVTemasL.ListItems(i).SubItems(4)
    .SubItems(3) = "Laboral"
   End With
  End If
 Next i
 For i = 1 To MainForm.LVTemasP.ListItems.Count
  If MainForm.LVTemasP.ListItems(i).SubItems(9) < 2 Then
   With LVTemasSinR.ListItems.Add(, , MainForm.LVTemasP.ListItems(i).SubItems(5))
    .SubItems(1) = MainForm.LVTemasP.ListItems(i).SubItems(1)
    .SubItems(2) = MainForm.LVTemasP.ListItems(i).SubItems(4)
    .SubItems(3) = "Personal"
   End With
  End If
  If DateDiff("d", MainForm.LVTemasP.ListItems(i).SubItems(5), Now) > 15 Then
   With LVTemas15d.ListItems.Add(, , MainForm.LVTemasP.ListItems(i).SubItems(5))
    .SubItems(1) = MainForm.LVTemasP.ListItems(i).SubItems(1)
    .SubItems(2) = MainForm.LVTemasP.ListItems(i).SubItems(4)
    .SubItems(3) = "Personal"
   End With
  End If
```

Módulos

A continuación se describirán los Módulos que proporcionan las funciones mencionadas en las descripciones anteriores para una mejor comprensión de su funcionamiento.
Tenemos dos módulos, el primero es el Módulo de Clase, en este se encuentran las funciones que van ligadas directamente a las funciones proporcionadas por *Windows*® para la manipulación de las ventanas y que básicamente se utilizan por el agente para determinar las ventanas abiertas, sus títulos y sus clases.
El segundo Módulo contiene funciones generales para la creación de las interfaces del programa, como son crear los calendarios, calcular la semana santa y la fase lunar, mostrar los eventos en el calendario, la función de búsqueda y el guardado de estadísticas.

Primeramente describiremos el Módulo de Clase.
Aquí tengo que hacer un paréntesis para informar al lector, que casi todas las funciones de este módulo fueron extraídas del sitio del famoso programador de *Visual Basic*® Guillermo Som, “El Guille”. Su código puede ser utilizado y/o modificado libremente siempre y cuando se haga mención de su autoría original.

En este módulo se utilizan 8 funciones de la API de *Windows*® y se explican brevemente a continuación, de acuerdo a las definiciones encontradas en la página de ayuda MSDN® de *Microsoft*©.

IsWindowVisible – Determina el estado de visibilidad de la ventana especificada. Recibe como único parámetro el hwnd (manejador) de la ventana a ser probada. Devuelve un valor diferente de cero si la ventana no está visible y cero en caso contrario.
GetWindowTextLength[7] – Devuelve la longitud, en caracteres, del texto de la barra de título de la ventana (si la ventana tiene barra de título). Si la ventana especificada es un control, entonces la función devuelve la longitud del texto dentro del control. Sin embargo, la función no puede devolver la longitud del texto de un control de edición en otra aplicación. Recibe como único parámetro el *hwnd* (manejador) de la ventana a ser probada. Si la función tiene éxito el valor devuelto es la longitud en caracteres del texto, si la ventana no tiene texto el valor devuelto es cero.
GetWindowText – Copia el texto de la barra de título de la ventana especificada (si tiene) en un búfer. Si la ventana especificada es un control, el texto del control es copiado. Sin embargo, la función no puede devolver el texto de un control en otra aplicación. Los parámetros son el manejador de la ventana que contiene el texto (*HWND*), el búfer que recibirá el texto. Si la cadena es más larga que el búfer la cadena es truncada y terminada con un caracter Null (LPTSTR), el máximo número de caracteres a copiar en el búfer, incluyendo el caracter Null, si el texto excede este límite es truncado (nMaxCount). Si la función tiene éxito el valor devuelto es el número de caracteres de la cadena copiada, sin incluir el caracter Null de terminación. Si la ventana no tiene barra de título o texto, si la barra de título está vacía o si la ventana o su manejador es inválido, el valor devuelto es cero.

GetDesktopWindow – Obtiene un manejador para la ventana activa o superior. La ventana activa es aquella que cubre la pantalla completa. La ventana activa es el área superior sobre la cual las otras ventanas son dibujadas. La función no tiene parámetros. El valor devuelto es un manejador a la ventana superior.

GetWindow – Obtiene un manejador para una ventana que tiene una relación específica (Z-Order o propietario) con la ventana específica. Como parámetros tiene un manejador a la ventana (el manejador obtenido es relativo a esta ventana, basado en el valor del parámetro uCmd) (*HWND*), la relación entre la ventana específica y la ventana cuyo manejador obtendrá la función (uCmd). Este último parámetro puede tener 7 diferentes valores de acuerdo a la relación. En nuestro caso, el valor buscado fue el 5, GW*_CHILD*, que identifica la ventana "hija" en la cima del Z-Order, si la ventana especificada es una ventana padre. De otra manera, el valor devuelto es cero. La función examina solamente las ventanas hijas de la ventana especificada. No examina ventanas "descendientes".
Si la función tiene éxito, el valor devuelto es un manejador de ventana. Si no existe ninguna ventana con la relación especificada con la ventana especificada, el valor devuelto es Null.

FindWindow – Obtiene un manejador a la ventana superior del Z-Order cuyo nombre de clase y de ventana coinciden con las cadenas especificadas. Esta función no busca ventanas hijas. Esta función no realiza búsquedas con diferenciación de mayúsculas. Los parámetros son el nombre de la clase (*lpClassName*) y el nombre de la ventana (el título de la ventana) (*lpWindowName*). Si el nombre de la clase se deja en null, encuentra cualquier ventana cuyo título coincida con el parámetro Nombre de la ventana. Si el nombre de la ventana se deja en Null entonces todos los nombres de las ventanas serán considerados.
Si la función tiene éxito, el valor devuelto es un manejador a la ventana que tiene el nombre de clase y el nombre de ventana especificados. Si la función falla, el valor devuelto será Null.

SendMessage – Manda el mensaje especificado a una ventana o ventanas. La función llama al procedimiento de la ventana para la ventana especificada y no regresa hasta que el procedimiento de la ventana ha procesado el mensaje. Los parámetros son el manejador de la ventana cuyo procedimiento recibirá el mensaje (si este parámetro es *HWND_BROADCAST* el mensaje es enviado a todas las ventanas superiores en el sistema, incluyendo ventanas deshabilitadas o

invisibles, traslapadas y emergentes, pero el mensaje no es enviado a ventanas hijas)(*Hwnd*), el mensaje a ser enviado (*Msg*), información adicional específica del mensaje (*wParam*), información adicional específica del mensaje (*lParam*). El valor devuelto por la función especifica el resultado del procesamiento del mensaje: *depende* del mensaje enviado.

GetClassName – Obtiene el nombre de la clase, a la cual pertenece la ventana especificada. Los parámetros son el manejador de ventana e indirectamente, la clase a la cual pertenece la ventana (*hWnd*), la cadena del nombre de la clase (*lpClassName*), el tamaño del buffer de *lpClassName* en caracteres; el buffer tiene que ser suficientemente largo para incluir el caracter de terminación Null, de otra manera la cadena de nombre de clase es truncada a *nMaxCount*-1 caracteres (*nMaxCount*).
El valor devuelto por la función, si tiene éxito, es el número de caracteres copiados al buffer, sin incluir el caracter de terminación *Null*. Si la función falla el valor devuelto es cero.

```
'------------------------------------------------------------------------------
'Clase para manipular ventanas de Windows                          (01/Ene/99)
'
'Esta clase enumera las ventas visibles, cierra la indicada, etc.
'
'©Guillermo 'guille' Som, 1999
'------------------------------------------------------------------------------
Option Explicit


'User32 Lib Apis


Private Declare Function IsWindowVisible Lib "user32" _
    (ByVal hwnd As Long) As Long
Private Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" _
    (ByVal hwnd As Long) As Long
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" _
    (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long

Private Declare Function GetDesktopWindow Lib "user32" () As Long

' GetWindow() Constants
Private Const GW_HWNDFIRST = 0&
Private Const GW_HWNDNEXT = 2&
Private Const GW_CHILD = 5&


Private Declare Function GetWindow Lib "user32" _
    (ByVal hwnd As Long, ByVal wFlag As Long) As Long

'
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" _
    (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
```

Una vez hechas estas definiciones, se explicará brevemente el uso de estas funciones dentro del Módulo de clase *cWindows*.
Primeramente en este módulo se declaran todas estas funciones con algunas de las constantes que se utilizarán también, mediante las instrucciones *Private Declare Function* y *Private Const*.
Ahora se crean funciones para usar las funciones de la API que se acaban de declarar.

WindowTitle.

Esta función devuelve el título de una ventana según el *hWnd* indicado que se recibe como parámetro. Se declaran 3 variables, para almacenar la longitud del título de la ventana convertido en cadena, la longitud del título y una variable que almacenará el texto del título de la ventana.
Mediante la función *GetWindowTextLength* se obtiene la longitud del título de la ventana a la que corresponde el *hWnd* y se almacena en lenTitulo. Si su valor es mayor de cero, entonces se incrementa en uno (por el carácter *Null*) y se convierte a cadena mediante la función *String*, almacenando el resultado en sTitulo. Con estos datos podemos llamar ahora a la función *GetWindowText* con el *hWnd* que se recibió como parámetro, la variable sTitulo como búfer con el tamaño de la cadena más uno, y con lenTitulo que ya contiene la longitud del título que será lo que leamos.
Finalmente la función nos devolverá en *WindowTitle* el Título completo de la ventana cuyo *hWnd* fue enviado como parámetro.

```
Private Function WindowTitle(ByVal hwnd As Long) As String
    'Devuelve el título de una ventana, según el hWnd indicado
    '
    Dim sTitulo As String
    Dim lenTitulo As Long
    Dim ret As Long

    'Leer la longitud del título de la ventana
    lenTitulo = GetWindowTextLength(hwnd)
    If lenTitulo > 0 Then
        lenTitulo = lenTitulo + 1
        sTitulo = String$(lenTitulo, 0)
        'Leer el título de la ventana
        ret = GetWindowText(hwnd, sTitulo, lenTitulo)
        WindowTitle = Left$(sTitulo, ret)
    End If
End Function
```

ClassName

Esta función devuelve la clase de la ventana cuyo título coincide con el enviado como parámetro. Se declaran tres variables *hWnd* para almacenar el manejador de la ventana a evaluar, *sClassName* que contendrá la cadena del nombre de clase de la ventana, y *nMaxCount* como máximo de caracteres de la cadena.
Para obtener el manejador de la ventana se utiliza la función *FindWindow*, mandando el título de la ventana cuya clase se desea obtener (que se envía como parámetro en esta función) y que como se mencionó algunas líneas arriba, cuando la función *FindWindow* se llama sin colocar la clase, busca la que coincida con el título enviado y devuelve el manejador *hWnd* de dicha ventana.
A continuación se establece la variable *nMaxCount* en 256 como longitud máxima de la clase de la ventana, y se reserva este espacio en el búfer donde se guardará la clase, mediante la función *Space* y como parámetro *nMaxCount*.
Ahora se hace una llamada a la función *GetClassName* con el *hWnd*, el búfer donde se guardará la clase (*sClassName*) y el máximo tamaño del búfer (*nMaxCount*) que también se tiene como variable que recibe el resultado de esta función.
Finalmente se le asigna a *ClassName* la cantidad de caracteres que se enviaron al búfer comenzando por la izquierda, mediante la función Left.

```
Public Function ClassName(ByVal Title As String) As String
    'Devuelve el ClassName de una ventana, indicando el título de la misma
    Dim hwnd As Long
    Dim sClassName As String
    Dim nMaxCount As Long

    hwnd = FindWindow(sClassName, Title)

    nMaxCount = 256
    sClassName = Space$(nMaxCount)
    nMaxCount = GetClassName(hwnd, sClassName, nMaxCount)
    ClassName = Left$(sClassName, nMaxCount)
End Function
```

Busca

La función *EnumTopWindows* que se explicará a continuación, necesita de una función que le facilite buscar una ventana, en el listado de ventanas que están abiertas, para decidir si es una ventana que ya se encontraba abierta o acaba de abrirse. Con esto se garantiza que la información respecto a las ventanas abiertas actualmente y abiertas durante el día de los datos de estadísticas sean reales. Dado que el monitoreo y búsqueda por nuevas ventanas se hace de manera constante (cada minuto) si esta función no existiera, la lista de ventanas abiertas sería interminable, pues cada minuto se agregarían todas las ventanas abiertas al listado sin importar que estuvieran repetidas o no.
La función simplemente compara la cadena que recibe como parámetro con cada uno de los elementos del listado que también recibe como parámetro, mediante un ciclo *For* que recorre el listado completo. Una vez que lo encuentra sale del ciclo y se le asigna al valor devuelto por la función el índice en el listado donde encontró la coincidencia. En caso de terminar el ciclo y no encontrar nada, se le asigna a este valor el número -1, indicando que el elemento no se encontró.

```
Public Function Busca(cad As String, Lista As ListView, Opcion As Integer) As Integer
Dim i

 'For i = 0 To Lista.ListCount - 1
 For i = 1 To Lista.ListItems.Count
  'If Lista.List(i) = Cad Then
  If Opcion = 1 Then
    If Lista.ListItems(i) = cad Then
     Busca = i
     Exit Function
    End If
   Else
    If Lista.ListItems(i).SubItems(3) = cad Then
     Busca = i
     Exit Function
    End If
   End If
 Next i
 Busca = -1
End Function
```

EnumTopWindows

Esta función, en las propias palabras del Guille, “Enumera las ventanas que tienen título y son visibles”.
La función básicamente lo que hace es revisar en todas las ventanas que se encuentran abiertas actualmente en el sistema y mediante la función buscar descrita previamente determina si la agrega o no en el listado de ventanas activas.
Primeramente se declaran las cinco variables que utilizaremos:

sTitulo – La cadena que contendrá el título de la ventana
hWnd – para el manejador de la ventana
col – una colección de objetos para las ventanas que se encuentren
Esta – una variable para guardar el valor devuelto por busca
contadorActivas - un entero para llevar el conteo de las ventanas activas

Lo primero que se hace es llamar a la función *GetWindow*, para determinar el manejador que se enviará como primer parámetro, se envía en su lugar la función *GetDesktopWindow*, que como se recordará devuelve el manejador de la ventana que se encuentra en la parte superior de la pila de ventanas mostradas en pantalla. Y como segundo parámetro se manda la constante *GW_CHILD*, para determinar si la ventana superior es una ventana padre. En caso de serlo la ventana es considerada en el conteo.
A continuación mediante un ciclo *Do While* iremos revisando ventana por ventana y continuará mientras el manejador devuelto por la función *GetWindow* sea diferente de cero. La diferencia es que ahora el segundo parámetro que se enviará será la constante *GW_HWNDNEXT* que buscará la siguiente ventana activa en el sistema devolviendo su *hWnd*.
Recordemos que este valor es único y es posible diferenciar a cada ventana con él. En otras palabras, no hay dos ventanas que tengan el mismo *hWnd*.

En el ciclo lo que se realiza es lo siguiente.
Primero determinamos si la ventana a la que hace referencia el manejador *hWnd* obtenido es visible o no, mediante la función *IsWindowVisible* descrita previamente. Si es así, obtenemos el Título de la ventana mediante la función *WindowTitle* y el manejador, y lo almacenamos en la variable sTitulo. Si la longitud del título es diferente de cero, entonces es hora de buscar el manejador en el *ListView* donde se están almacenando las ventanas actuales y que se envía como parámetro en la función principal EnumTopWindows.
Llamamos a la función Busca con estos parámetros y almacenamos el valor devuelto en la variable Esta.
Si la variable Esta es -1, entonces no se encontró el *hWnd* y tenemos que agregarlo al *ListView*. Antes de agregarlo nos aseguramos que no se trate de la ventana del programa en sí, pues nos interesan solamente los otros programas que el usuario está empleando. Para hacerlo, simplemente se necesita revisar el valor de sTítulo y si es diferente a “Saturnus” y “*Program Manager*” (por el editor de *Visual Basic*® para las pruebas) entonces agregamos los siguientes datos en el *ListView*:
El título de la ventana, la clase de la ventana (que obtenemos llamando a la función *ClassName* y como parámetro el título de la ventana) y el manejador *hWnd*. También establecemos en cero el valor del contador de ventanas repetidas.
Entonces cuando la función Busca nos devuelve un valor de 1, lo que hacemos es incrementar este valor de la columna en uno.
Existe también una variable, ContadorActivas que lleva el conteo de las ventanas que se encuentran actualmente abiertas y que será mostrado en la ventana del resumen.

```
Public Function EnumTopWindows(Optional unListView As ListView) As Variant
    Dim sTitulo As String
    Dim hwnd As Long
    Dim col As Collection
    Dim Esta As Integer
    Dim ContadorActivas As Integer
    Set col = New Collection
    'Primera ventana
    hwnd = GetWindow(GetDesktopWindow(), GW_CHILD)

    'Recorrer el resto de las ventanas
    Do While hwnd <> 0&
        'Si la ventana es visible
        If IsWindowVisible(hwnd) Then
            'Leer el caption de la ventana
            sTitulo = WindowTitle(hwnd)
            If Len(sTitulo) Then
                'Añadimos el título
                col.Add sTitulo
                'y el hWnd por si fuese útil
                col.Add hwnd
                'Si se especifica el ListBox
                If Not unListView Is Nothing Then
                  Esta = Busca(Trim(Str(hwnd)), unListView, 3)
                  If Esta = -1 Then
                   If sTitulo <> "Saturnus" And sTitulo <> "iRobot" And sTitulo <> "Program Mana
                    With unListView.ListItems.Add(, , sTitulo)
                        .SubItems(1) = 0
                        .SubItems(2) = ClassName(sTitulo)
                        .SubItems(3) = hwnd
                    End With
                    End If
                  Else
                     With unListView.ListItems(Esta)
                       .SubItems(1) = .SubItems(1) + 1
                     End With
```

Ahora describiremos el módulo 1 que es el que contiene procedimientos y funciones adicionales que facilitan el procesamiento en las interfaces del programa.

El primer procedimiento que se describirá es de los más importantes en este módulo y es quien hace uso de otros que se encuentran en el mismo módulo y que se describirán más adelante.

CreaCal

Este procedimiento se encarga de “construir” la vista mensual del calendario, con los días en el orden correcto, algunos días previos y posteriores de los meses previos y posteriores respectivamente, los colores según se trate de días ya pasados o por venir, y aquellos que tienen algún evento relacionado. Entre otras cosas más.
Los parámetros que recibe este procedimiento son DiaCal (que contiene el día actual a mostrar), MesCal (el mes que se desplegará), AñoCal (el año del mes a desplegar) y DomR (que corresponde al domingo de resurrección de la semana santa del año a mostrar).
Existen dos variables globales llamadas MesCalAct y AñoCalAct que contienen el mes actual del calendario o mes mostrado en la interfaz, y el año del mismo respectivamente. Esta información es útil para cuando es necesario mostrar (normalmente regresar) al calendario actual activo, pero también es útil para otras cuestiones que veremos posteriormente.
Entonces como primer paso en el procedimiento, lo que se hace es almacenar los valores de mes y año que entran como parámetros en dichas variables.

A continuación, de acuerdo al mes con un *Select Case*, se coloca el valor de la etiqueta del mes desplegado (LCalM) concatenada con el año en la variable AñoCal y se guarda en la variable totaldm el número de días del mes correspondiente. Para el caso de febrero, es necesario calcular si AñoCal es un año bisiesto o no, lo cual se hace mediante una división exacta entre 400, si el año es múltiplo de 400, entonces se trata de un año bisiesto y totaldm tendrá 29, de lo contrario será 28.

El domingo de resurrección se calcula mediante una función llamada CalculaSemanaSanta que devuelve una cadena con el día y mes separados por una "/". Esta función se explicará más adelante.
Recuerde que como parámetro del procedimiento que estamos analizando se recibe el domingo de resurrección (DomR), que se obtiene de la función mencionada previamente. Eso quiere decir que el formato de este parámetro es dd/mm. Mediante funciones de cadenas obtenemos los valores que están antes y después de la "/" y los almacenamos en las variables DiaSS y MesSS.
Con estos datos, ya es posible llamar a la función CargaEventos, que lleva como parámetros, el día, mes y año a cargar, así como el día y el mes de la semana santa que ya tenemos en las variables DiaSS y MesSS.

En este procedimiento tenemos una variable llamada DiaBase, que contendrá el día de la semana que corresponde al primer día del mes y año a desplegar. Al día domingo le corresponde el 1, al lunes el 2 y así sucesivamente. Esta variable es muy importante pues a partir de ella es posible construir correctamente el calendario mensual como veremos posteriormente.
Es hora de colocar los días en las etiquetas así como los colores correspondientes.
Mediante un ciclo *For…Next* que va desde el día uno hasta el último día del mes (totaldm), iremos colocando cada uno de los días en etiquetas.

Estas etiquetas forman parte de una matriz llamada LCalD y para colocar el día correcto se utiliza la fórmula (i + DíaBase – 2). Donde i corresponde al día a mostrar y DiaBase el día de la semana del primer día del mes. Por ejemplo, si el primer día del mes fue jueves entonces el valor de DiaBase será 5, y hoy es 17 (el valor de i) entonces, el índice en la matriz de etiquetas LCalD que le corresponde sería 17+5-2=20.

Los primeros en colocarse serán los días del mes previos al día de hoy. Al entrar al ciclo se hace una comparación y si la i es menor al día de hoy y el mes y año son los actuales entonces el color de la etiqueta será un gris oscuro y el fondo es negro. Si no es así, entonces quiere decir que son los días posteriores a hoy, y por lo tanto irán en un color gris claro. El fondo también se coloca en negro. Este detalle es importante pues si en el mes desplegado previamente el día actual corresponde a otro elemento de la matriz de etiquetas (que es lo más común) entonces el fondo estará colocado en color azul y es necesario restablecerlo al color negro de fondo.

```
Public Sub CreaCal(DiaCal As Integer, MesCal, AñoCal As Integer, DomR As String)
Dim DiaBase, totaldm, totalda, DiasAnt, DiaSS As Integer, MesSS As Integer

 MesCalAct = MesCal
 AñoCalAct = AñoCal
 Select Case MesCal
       Case 1: MainForm.LCalM.Caption = "Enero " & AñoCal: totaldm = 31: totalda = 0: DiasAnt = 3
       Case 2: MainForm.LCalM.Caption = "Febrero " & AñoCal: If AñoCal Mod 400 = 0 Then totaldm =
       Case 3: MainForm.LCalM.Caption = "Marzo " & AñoCal: totaldm = 31: totalda = 59: If AñoCal
       Case 4: MainForm.LCalM.Caption = "Abril " & AñoCal: totaldm = 30: totalda = 90: DiasAnt =
       Case 5: MainForm.LCalM.Caption = "Mayo " & AñoCal: totaldm = 31: totalda = 120: DiasAnt =
       Case 6: MainForm.LCalM.Caption = "Junio " & AñoCal: totaldm = 30: totalda = 151: DiasAnt =
       Case 7: MainForm.LCalM.Caption = "Julio " & AñoCal: totaldm = 31: totalda = 181: DiasAnt =
       Case 8: MainForm.LCalM.Caption = "Agosto " & AñoCal: totaldm = 31: totalda = 212: DiasAnt
       Case 9: MainForm.LCalM.Caption = "Septiembre " & AñoCal: totaldm = 30: totalda = 243: Dias
       Case 10: MainForm.LCalM.Caption = "Octubre " & AñoCal: totaldm = 31: totalda = 273: DiasAn
       Case 11: MainForm.LCalM.Caption = "Noviembre " & AñoCal: totaldm = 30: totalda = 304: Dias
       Case 12: MainForm.LCalM.Caption = "Diciembre " & AñoCal: totaldm = 31: totalda = 334: Dias
 End Select
DiaSS = Val(Left(DomR, InStr(1, DomR, "/")))
MesSS = Val(Right(DomR, Len(DomR) - InStr(1, DomR, "/")))
CargaEventos DiaCal, UCase(Left(MainForm.LCalM.Caption, 3)), AñoCal, DiaSS, MesSS
DiaBase = Weekday("01/" & MesCal & "/" & AñoCal)
For i = 1 To totaldm
  pos = InStr(1, MainForm.DiasEventos, "$" & i & "$")
  If pos <> 0 Then
  End If
 MainForm.LCalD(i + DiaBase - 2).Caption = i
 If i < Day(Date) And MesCal = Month(Date) And AñoCal = Year(Date) Then
  MainForm.LCalD(i + DiaBase - 2).ForeColor = &H606060 'días pasados del mes
  MainForm.LCalD(i + DiaBase - 2).BackColor = &H0
 Else
  MainForm.LCalD(i + DiaBase - 2).ForeColor = &H909090 'días siguientes del mes
  MainForm.LCalD(i + DiaBase - 2).BackColor = &H0
  If i < 10 Then
   If InStr(1, MainForm.DiasEventos, "$0" & i & "$") <> 0 Then MainForm.LCalD(i + DiaBase - 2).Fo
```

En la función de CargaEventos que se describirá posteriormente, se crea una cadena que almacena los días del mes que tienen eventos, separados por un caracter “$”. Esta cadena es una variable global del *MainForm* llamada DiasEventos.
Entonces, mediante la función Instr y concatenando al valor de i el caracter “$” antes y después de i podemos saber si el día que estamos por colocar en el calendario tiene un evento o no, y por tanto, en caso afirmativo, cambiamos el color de la fuente (*Forecolor*) a un color azul.
Un detalle en particular antes de buscar el valor de i en la cadena de eventos, es si aquél es menor de diez, entonces es necesario colocar un cero antes pues así es como se almacena en la cadena y de lo contrario no podría ser localizado. Esto se sabe mediante un condicional sencillo.
Ahora, para el día actual se tiene una consideración especial, que es colocar el fondo en color gris si el día no tiene evento. Es decir, si el valor de i es igual al día actual y el mes y año también, entonces el estilo de la etiqueta se modifica de transparente a sólido para que pueda mostrarse en color, el *Forecolor* se modifica a un gris muy claro, y el fondo en un gris ligeramente claro.
Como es posible que el día actual tenga también un evento, entonces debe buscarse en la cadena de eventos (aplica el mismo detalle mencionado anteriormente de colocarle un cero si es menor de diez) y en caso de estar, entonces el color del fondo no será gris, sino azul.
Este proceso se repite con todos los días del mes.
Para finalizar este procedimiento únicamente falta considerar los últimos días del mes anterior y los primeros del mes siguiente.
Para ello haremos dos ciclos, el primero será para los últimos del mes anterior. Para calcular los límites del ciclo se toma como inicio el último elemento marcado más uno, es decir totaldm + DiaBase – 2 +1, o lo que es lo mismo totaldm + DiaBase – 1. El final del ciclo será el último elemento de la matriz que es el 41.
Dentro del ciclo solamente se cambiará el Forecolor de las etiquetas a un gris oscuro y para el valor de la etiqueta se utiliza la fórmula i - (totaldm + DiaBase - 2).

Para el otro ciclo, los límites serán el primer elemento de la matriz, es decir, cero. Y como final será DiaBase – 2. La lógica nos diría que debería ser DiaBase – 1, sin embargo, dado que el primer índice de la matriz es cero, entonces debe restarse siempre uno, al valor requerido.

```
   If i < 10 Then
    If InStr(1, MainForm.DiasEventos, "$0" & i & "$") <> 0 Then MainForm.LCalD(i + DiaBase - 2).Fo
   Else
    If InStr(1, MainForm.DiasEventos, "$" & i & "$") <> 0 Then MainForm.LCalD(i + DiaBase - 2).For
   End If
  End If
 'día actual
  If i = Day(Date) And MesCal = Month(Date) And AñoCal = Year(Date) Then
   MainForm.LCalD(i + DiaBase - 2).ForeColor = &HDDDDDD
   MainForm.LCalD(i + DiaBase - 2).BackStyle = 1
   MainForm.LCalD(i + DiaBase - 2).BackColor = &H404040
   If i < 10 Then
    If InStr(1, MainForm.DiasEventos, "$0" & i & "$") <> 0 Then MainForm.LCalD(i + DiaBase - 2).Ba
   Else
    If InStr(1, MainForm.DiasEventos, "$" & i & "$") <> 0 Then MainForm.LCalD(i + DiaBase - 2).Bac
   End If
  End If
 Next i

 'dias restantes del mes anterior
 For i = (totaldm + DiaBase - 1) To 41
  MainForm.LCalD(i).ForeColor = &H202020
  MainForm.LCalD(i).Caption = i - (totaldm + DiaBase - 2)
 Next i
 'primeros días del mes siguiente
 For i = 0 To DiaBase - 2
   MainForm.LCalD(i).ForeColor = &H202020
   MainForm.LCalD(i).Caption = (DiasAnt - DiaBase + 2) + i
 Next i
 End Sub
```

CargaEventos

El segundo procedimiento se denomina CargaEventos y como su nombre lo indica, se encarga principalmente de leer los eventos del archivo de texto correspondiente y mostrarlos en el calendario.
Los parámetros de esta función son Dia, Mes, Año y como opcionales el día de semana santa (DSS) y mes de semana santa (MSS), estas dos últimas se utilizan para facilitar el despliegue de la información de los días jueves y viernes santos en la marquesina y en el calendario si el mes mostrado corresponde al de la semana santa en ese año.
Para esto tenemos dos variables adicionales en el procedimiento.
MssLet, cuyo valor corresponde a las primeras tres letras de los meses Marzo y Abril según corresponda ese año al mes de la semana santa y que es asignado al momento de revisar si el parámetro de entrada MSS es 3 o 4 respectivamente.
Una vez asignado el valor a la variable, lo comparamos con el valor mes, que se recibe también en el procedimiento como parámetro.
Si coinciden, entonces hacemos uso de la otra variable que tenemos que es AgregarSS, cuyo valor booleano será *True* si se da esta coincidencia, *False* si no, y cuyo uso veremos más adelante.

Ahora definimos las rutas de los archivos Eventos y EventosRep que ya fueron descritos previamente, y para ellos se utilizan dos variables locales NombreEventos y NombreEventosRep.
Para formar el nombre del archivo simplemente concatenamos la variable global Ruta, con “EventosRep.txt” para los eventos repetitivos y con “Eventos” más el parámetro de entrada “Año” y “.txt” para abrir el archivo de eventos del año a desplegar.
El primer archivo a revisar es el de eventos periódicos o repetitivos. En una variable llamada Cad, verificamos mediante la función Dir si el archivo de eventos periódicos existe, si la variable recibe una cadena vacía, el archivo no existe. Si existe, entonces abrimos el archivo mediante la función *Open…For Input*.

Reiniciamos la variable Cad a “” porque será utilizada posteriormente, y vamos leyendo línea a línea el archivo almacenando la línea leída en la variable v0.
Los primeros 3 caracteres de la cadena corresponden a las tres primeras letras del mes, entonces comparamos estos 3 caracteres con el parámetro de entrada mes y si coincide (hay evento guardado para ese mes) entonces recortamos esos 3 caracteres más un guión de la variable v0 y continuamos leyendo información de esta variable. Leemos los siguientes dos caracteres que coinciden ahora con el día del evento y dado que vamos a colocarlo en el calendario mostrado solamente nos interesan los eventos que son posteriores al día de hoy; mediante una comparación determinamos si esto es así y entonces a la variable Cad, agregaremos el valor de v0 que será el día y el texto del evento separados por un guión. Esta variable Cad será asignada posteriormente a la marquesina. O sea que todo lo que se vaya concatenando a ella será desplegado en la marquesina al final del procedimiento.
Es necesario agregar también el día del evento en la cadena global DiasEventos pues como se recordará, esta variable es utilizada por el procedimiento que dibuja o “construye” el calendario para marcar con un color distinto los días que tienen evento.
Para hacerlo simplemente tomamos los caracteres que se encuentran antes del guión en la variable v0 y los concatenamos a la cadena DiasEventos, colocando un signo “$” antes del número. Podríamos simplemente tomar los dos primeros caracteres de la variable, sin embargo, hay días que solo son de un dígito y eso nos devolvería información errónea (incluiría el guión).

```
Public Sub CargaEventos(dia As Integer, mes As String, Anio As Integer, Optional DSS As Integer,
Dim Nombre, MasEventos, AgregarSS As Boolean, MssLet
If MSS = 3 Then MssLet = "MAR"
If MSS = 4 Then MssLet = "ABR"
If mes = MssLet Then
  AgregarSS = True
End If

NombreEventos = MainForm.Ruta & "Eventos" & Anio & ".txt"
NombreEventosRep = MainForm.Ruta & "EventosRep.txt"
'************************* Eventos periódicos
cad = Dir(NombreEventosRep)
MainForm.DiasEventos = ""
If cad <> "" Then
 Open NombreEventosRep For Input As #1
 Fila = 0
 cad = ""
 Do While Not EOF(1)
  Input #1, v0
   If Left(v0, 3) = mes Then
    v0 = Right(v0, Len(v0) - 4)
    If Val(Left(v0, 2)) >= dia Then
     cad = cad & " | " & v0
     pos = InStr(1, v0, "-")
     MainForm.DiasEventos = MainForm.DiasEventos & "$" & Left(v0, pos - 1)
     MasEventos = False
    Else
     MasEventos = True
    End If
   End If
 Loop
 Close #1
 MainForm.Lmsg.Caption = cad
 If cad <> "" Then MasEventos = True
 MainForm.DiasEventos = MainForm.DiasEventos & "$"
End If
```

También existe una variable llamada MasEventos que se utiliza para colocar una nota en la marquesina al usuario e indicarle que ese mes tiene más eventos pero son previos a la fecha actual. Es decir, tomará el valor *True*, al cumplir la condición de estar en el mes, pero ser menor al día de hoy y *False* al encontrarse en un día mayor o igual a hoy.
Después de cerrar el ciclo de lectura del archivo, éste se cierra mediante la instrucción *Close*.

En la marquesina se coloca el valor de la variable Cad y como medida adicional, en caso de que Cad se encuentre vacía, entonces la variable MasEventos se establece en *True*.
Finalmente la variable global DiasEventos se cierra colocando un signo “$” al final.

Ahora toca el turno al archivo de eventos normales o no repetitivos. El procedimiento es similar al anterior, salvo algunas excepciones.

Lo primero nuevamente es revisar si el archivo de texto existe, empleando la función *Dir*. En el caso de que dicha función devuelva un valor vacío, entonces el archivo no existe y colocaremos en la marquesina el texto “No existe el archivo de eventos“ y el año que corresponda.
Sin embargo, antes de salir del condicional, necesitamos agregar el jueves y viernes santos en el caso de que correspondan al mes que estamos desplegando. Esto lo comprobamos al revisar la variable AgregarSS que tendrá *True* en caso afirmativo. El parámetro de entrada DSS contiene el día del jueves santo, entonces bastará con agregar a la cadena CAD el DSS más la descripción “-Jueves Santo” y el DSS+1 con la descripción “-Viernes Santo”.

También es necesario que en la variable global DiasEventos coloquemos DSS y DSS + 1. Para ambos casos solamente se agregará DSS + 1 si DSS es diferente de 31, pues indicaría que se encuentra al fin del mes de Marzo y sumarle uno sería erróneo.
Así, actualizamos el valor de la marquesina y finalmente cerramos la variable DiasEventos con un signo “$”.
En el caso de que el archivo de eventos del año sí exista, entonces leeremos línea a línea el archivo, almacenándolas en la variable v0. Verificamos de esa variable que los tres primeros caracteres correspondan al mes desplegado, confirmamos si el evento es en un día anterior o posterior al día de hoy, si es posterior lo agregamos a la cadena Cad y a la variable DiasEventos separado con signo “$” modificamos la variable MasEventos según sea el caso y cerramos el archivo, todo de la misma manera como se explicó previamente para los eventos periódicos o repetitivos.

Al igual que como cuando el archivo de eventos no existe, también debemos verificar el valor de la variable AgregarSS. Si es *True*, agregamos el jueves y viernes santos como se describió líneas arriba.

```
      MainForm.DiasEventos = MainForm.DiasEventos & "$" & DSS
      If DSS <> 31 Then MainForm.DiasEventos = MainForm.DiasEventos & "$" & DSS + 1
      MainForm.Lmsg.Caption = MainForm.Lmsg.Caption & cad
      MainForm.DiasEventos = MainForm.DiasEventos & "$"
    End If
Else
 Open NombreEventos For Input As #1
 Fila = 0
 cad = ""
 Do While Not EOF(1)
  Input #1, v0
   If Left(v0, 3) = mes Then
    v0 = Right(v0, Len(v0) - 4)
    If Val(Left(v0, 2)) >= dia Then
     cad = cad & " | " & v0
     pos = InStr(1, v0, "-")
     MainForm.DiasEventos = MainForm.DiasEventos & "$" & Left(v0, pos - 1)
     MasEventos = False
    Else
     MasEventos = True
    End If
   End If
 Loop
 Close #1
If AgregarSS = True Then
  cad = cad & " | " & DSS & "- Jueves Santo"
  If DSS <> 31 Then cad = cad & " | " & DSS + 1 & "- Viernes Santo"
  MainForm.DiasEventos = MainForm.DiasEventos & "$" & DSS
  If DSS <> 31 Then MainForm.DiasEventos = MainForm.DiasEventos & "$" & DSS + 1
End If
If cad = "" Then cad = " | No hay eventos para este mes."
 If MasEventos = True Then cad = " | No hay más eventos para este mes."
 MainForm.Lmsg.Caption = MainForm.Lmsg.Caption & cad
 MainForm.DiasEventos = MainForm.DiasEventos & "$"
End If
```

MuestraFaseLunar

Este procedimiento realiza el cálculo de la fase lunar que corresponde a la fecha recibida en los parámetros como día, mes y año.
El código del procedimiento fue extraído de la página:

http://www.experts-exchange.com/Programming/Languages/Scripting/PHP/Q_26472761.html[11]

donde se encuentra un script en PHP. Simplemente se hizo la traducción de PHP a *Visual Basic*® y se adecuó para que de un objeto *ImageList* llamado LunarPics, se obtenga la imagen de la fase lunar a mostrar. El código utiliza valores y constantes ya definidas que no se explicarán aquí, pues desconozco el origen de las mismas. Baste con saber que el cálculo de las fases es correcto en las pruebas que se han realizado con fechas aleatorias desde 1900 y hasta 2150.
El resultado de los cálculos es un valor de 0 a 7, cada uno identificando una fase lunar distinta: 0-Luna nueva, 1-Poco creciente, 2-Cuarto creciente, 3-Muy creciente, 4-Luna llena, 5-Poco menguante, 6-Cuarto menguante, 7-Muy menguante.
Además, para dar una información más precisa, en el *ToolTipText* de la imagen de la fase lunar que se muestra en un objeto *Picture* llamado PLPhase, se muestra un número que indica el porcentaje de la edad lunar respecto a su fase mensual, donde 100 corresponde a la luna nueva y 50 a la luna llena.

```php
<?php
function moon_phase($year, $month, $day)
{
        $c = $e = $jd = $b = 0;
        if ($month < 3)
        {
                $year--;
                $month += 12;
        }
        ++$month;
        $c = 365.25 * $year;
        $e = 30.6 * $month;
        $jd = $c + $e + $day - 694039.09; //jd is total days elapsed
        $jd /= 29.5305882; //divide by the moon cycle
        $b = (int) $jd; //int(jd) -> b, take integer part of jd
        $jd -= $b; //subtract integer part to leave fractional part of original
        $b = round($jd * 8); //scale fraction from 0-8 and round
        if ($b >= 8 )
        {
                $b = 0;//0 and 8 are the same so turn 8 into 0
        }
        switch ($b)
        {
                case 0:
                        return 'New Moon';
                        break;
                case 1:
                        return 'Waxing Crescent Moon';
                        break;
                case 2:
                        return 'Quarter Moon';
                        break;
                case 3:
                        return 'Waxing Gibbous Moon';
                        break;
                case 4:
                        return 'Full Moon';
```

CalculaSemanaSanta

Esta función también fue extraída de internet, código abierto, en este caso de un Algoritmo del calendario eclesiástico de *Butcher*, o simplemente algoritmo de *Butcher* y que se encuentra en la siguiente página:

http://www.smart.net/~mmontes/nature1876.html

Nuevamente, se hicieron ajustes para adaptar el resultado con lo que necesitamos para el calendario, pues la función nos entrega el domingo de pascua, y nosotros necesitamos conocer tanto el jueves como el viernes santo. Este último lo calculamos sumando uno al jueves, si es diferente de 31 como se mencionó previamente.
Entonces, simplemente realizamos algunas condicionales para asegurar que el día sea correcto según el mes. Por ejemplo si el resultado de los cálculos nos arroja un 1 de abril, entonces el jueves santo deberá ser el 29 de marzo, y es lo que devolverá la función, 29/3.

Nature (1876) Algorithm for Calculating the Date of Easter in the Gregorian Calendar

Originally published 1996 December by M.J. Montes.
This page was last updated 2001 July 31 by Marcos J. Montes.

The actual origin of this algorithm appears to be by an anonymous correspondent from New York to *Nature* in 1876 (*Thanks Denis!*). Samuel Butcher, Bishop Meath, showed that this algorithm followed from Delambre's analytical solutions, and produces the date of Easter for all years. You can see the algorithm, as we version for Orthodox Easter at another Easter Date site.

This algorithm appears in *Practical Astronomy With Your Calculator, 2nd Edition* by Peter Duffett-Smith, and he obtained this algorithm from *Butcher's Ecclesiastical Calendar*, published in 1876. This algorithm has also been published in the 1922 book *General Astronomy* by Spencer Jones; in *The Journal the British Astronomical Association* (Vol.88, page 91, December 1977); and in *Astronomical Algorithms* (1991) by Jean Meeus.

This algorithm holds for any year in the Gregorian Calendar, which (of course) means years including and after 1583.

In the text below, / represents an integer division neglecting the remainder, while **%** is division keeping only the remainder. So 30/7=4 , and 30%7=2 .

```
a=year%19
b=year/100
c=year%100
d=b/4
e=b%4
f=(b+8)/25
g=(b-f+1)/3
h=(19*a+b-d-g+15)%30
i=c/4
k=c%4
l=(32+2*e+2*i-h-k)%7
m=(a+11*h+22*l)/451
Easter Month =(h+l-7*m+114)/31  [3=March, 4=April]
p=(h+l-7*m+114)%31
Easter Date=p+1     (date in Easter Month)
```

Busca

Este procedimiento es muy importante en las interfaces de Contactos, Citas y Notas, pues facilita la búsqueda de los datos al teclear en los campos de texto. Dado que se trata de la misma función en tres interfaces, se adaptó para ser un procedimiento reusable colocado en el módulo. Simplemente fue necesario colocar los objetos *ListView* como parámetros de entrada y obviamente las cadenas a buscar. Sin embargo hubo un detalle que complicaba la búsqueda, pues debe saberse si se trata de una búsqueda en la propiedad Texto del *ListView* (columna principal) o en los *Subitems* (1..n); por lo tanto, fue necesario incluir en los parámetros de entrada una variable booleana llamada itText y otras dos enteras TotalSI y SI. El parámetro itText nos dice si se trata del texto principal del *ListView*, mientras que TotalSI nos indica el total de *Subitems* del *ListView* y SI el *Subitem* en el que realizaremos la búsqueda.
Recuerde que en la descripción de las interfaces se mencionó que existe en la interfaz principal un *ListView* que se utiliza exclusivamente para mostrar los resultados de las búsquedas en las demás interfaces, este objeto se llama LVBusqueda.
El primer paso que debemos realizar es limpiar el contenido de LVBusqueda pues podría contener información de la última búsqueda realizada.
Posteriormente se hace un ciclo para recorrer todo el *ListView* que entra como parámetro, Lista. Y en el caso de que el parámetro itText sea verdadero, entonces buscaremos la cadena Cad, convertida a mayúsculas, en la propiedad Text de cada elemento de Lista, mediante la función InStr que será la que realiza la búsqueda. Si lo encuentra, entonces agregamos en LVBusqueda este *Ítem* y todos los *Subitems* que contenga.
Adicionalmente guardamos el índice (i) del elemento encontrado en la propiedad *Tag* del elemento del *ListView*. Con el fin de que cuando se seleccione en el LVBusqueda de la interfaz podamos localizarlo en el *Listview* original sin problemas.

Para el caso de que ItText sea falso, es decir que la búsqueda se realiza en alguno de los S*ubitems*, entonces el proceso es el mismo excepto para la función *InStr* buscará no en la propiedad Text del *ListView*, sino en el *Subitem* indicado con el parámetro SI.
Todo el resto es exactamente igual.

```
Public Sub Busca(cad As String, Lista As ListView, ItText As Boolean, Optional TotalSI As Integer
Dim i, j
MainForm.LVBusqueda.ListItems.Clear
For i = 1 To Lista.ListItems.Count
 If ItText = True Then
  If InStr(1, UCase(Lista.ListItems(i).Text), UCase(cad)) <> 0 Then
   With MainForm.LVBusqueda.ListItems.Add(, , Lista.ListItems(i).Text)
    For j = 1 To TotalSI
    .SubItems(j) = Lista.ListItems(i).SubItems(j)
    Next j
   End With
   MainForm.LVBusqueda.ListItems.Item(MainForm.LVBusqueda.ListItems.Count).Tag = i
  End If
 Else
  If InStr(1, UCase(Lista.ListItems(i).SubItems(SI)), UCase(cad)) <> 0 Then
   With MainForm.LVBusqueda.ListItems.Add(, , Lista.ListItems(i).Text)
    For j = 1 To TotalSI
    .SubItems(j) = Lista.ListItems(i).SubItems(j)
    Next j
   End With
   MainForm.LVBusqueda.ListItems.Item(MainForm.LVBusqueda.ListItems.Count).Tag = i
  End If
 End If
Next i
End Sub
```

IsFormLoaded

Esta función, como se recordará, es utilizada para buscar el formulario que avisa al usuario el tiempo que ha estado sin inactividad y le sugiere actividades a realizar.
Para determinar si el formulario está cargado, simplemente deben recorrerse todos los formularios mediante el objeto *Forms* y su propiedad *Count* en un ciclo.
Como parámetro de la función recibimos únicamente el nombre del formulario y como resultado devolvemos un *True* o *False*.
Si el nombre del formulario coincide con alguno de los formularios que se recorren durante el ciclo, entonces devolvemos el *True* y salimos de la función.

```
Function IsFormLoaded(FormToCheck As Form) As Integer
Dim Y As Integer

For Y = 0 To Forms.Count - 1
If Forms(Y) Is FormToCheck Then
    IsFormLoaded = True
    Exit Function
    End If
Next
IsFormLoaded = False
End Function
```

GuardaEst

Finalmente tenemos este procedimiento que se encarga de almacenar en un archivo los datos del formulario FEstAct y específicamente del *Listview* LVResumen.
Todos los *Subitems* son guardados en el archivo cada media hora y también cuando el usuario cierra el programa.
Estos datos son recuperados cuando se abre el formulario FEstAct como se detallo previamente.

```
Public Sub GuardaEst()
Dim Arch

Arch = MainForm.Ruta & "EstAct.txt"
 Open Arch For Output As #1
   For i = 1 To FEstAct.LVResumen.ListItems.Count
    Write #1, Format(FEstAct.LVResumen.ListItems(i).Text, "mmm-dd-yyyy"), FEstAct.LVResumen.ListI
    FEstAct.LVResumen.ListItems(i).SubItems(3), FEstAct.LVResumen.ListItems(i).SubItems(4), FEstA
    FEstAct.LVResumen.ListItems(i).SubItems(6), FEstAct.LVResumen.ListItems(i).SubItems(7), FEstA
    FEstAct.LVResumen.ListItems(i).SubItems(9)
   Next i
 Close #1

End Sub
```

Estas son todas las funciones y procedimientos almacenados en el Módulo 1 que nos ayudarán a ejecutar todas las acciones y tareas del programa de manera óptima y organizada.

Pruebas y Correcciones

Durante el proceso de creación del programa se fueron desarrollando diferentes versiones como se comentaba en los primeros capítulos, pues dada la naturaleza del programa y del área de aplicación (oficina) era necesario que los usuarios experimentaran con las interfaces para determinar si éstas eran suficientes y necesarias para sus fines y habilidades.
En las primeras versiones del programa la interfaz contenía mucha información en una sola ventana y puesto que la idea del mismo es tener un reloj en segundo plano que no interfiera con el resto de las ventanas y actividades del usuario, poco a poco se fueron reduciendo los contenidos de las ventanas hasta llegar a un diseño minimalista que muestra y solicita solo la información que es absolutamente necesaria.

En el caso de las agendas, se tuvieron que realizar algunas reuniones con los coordinadores de los grupos para poder determinar la información que se requería registrar y de manera que todos los grupos involucrados pudieran hacerlo. Ya que cada grupo maneja distinta información, fue necesario también hacer un consenso para agrupar los campos y botones compartidos.
Finalmente se definieron cinco clasificaciones: *Serie, Cambio de Modelo, Proyecto, KÜ y DLV.*
Y nueve actividades: TUL, Datenübergabe, Aclaraciones, AWS, Reporte JIT, Plan de Plazos, Juntas, Reportes y Otros.
Por razones de confidencialidad, no se describirán los significados de estas clasificaciones y actividades, sin embargo, sí puede confirmarse que dichas opciones fueron suficientes para abarcar todas las actividades medibles que se desarrollan en el departamento.
Para la Agenda personal, la clasificación final quedó acordada de la siguiente manera: Familia, Amigos, Hobbies, Profesional, Otros. De la misma manera que la agenda laboral, se realizaron reuniones y consensos para llegar a la conclusión descrita. Había muchas más clasificaciones que podían incluirse, y algunas fueron derivadas al área de actividades que quedó como sigue: *Internet, Llamada, Pagos, Trámites, Alimentos, Lectura, Compras, Estudio, Salud, Otros.*
Estas clasificaciones y actividades son meramente informativas, y aunque podrían considerarse para un estudio más detallado en las interfaces de Reportes, esto se hará posterior a este documento por razones de tiempo. Los usuarios estuvieron de acuerdo con esta decisión pues por el momento solo les interesa medir tiempos de actividades laborales y personales en general.

Para cronometrar el tiempo, en las primeras versiones no se habían incluido los botones de *Play*, *Pausa* y *Stop*. Únicamente había un botón gigante de Iniciar que se transformaba en Detener y aunque cumplía los propósitos no era estéticamente agradable al diseño minimalista del programa.

La marquesina no aparecía en las primeras versiones, pero un colega mencionó que en los noticieros y pantallas gigantes de la ciudad le parecía una excelente opción el poder ver el resumen de la información desplazarse constantemente para no perder detalle y leerlo rápidamente. Esto generó la idea de mostrar la información relevante a manera de marquesina en

un espacio especial diseñado para ello. Además de ser un área muy dinámica y práctica para proporcionar al usuario información adicional sobre los archivos, instrucciones y problemas encontrados.

En los contactos, inicialmente se había colocado textos describiendo los campos, pero esto restaba espacio a la información además de que para algunos era ilegible por el tamaño tan pequeño de la fuente. Es por eso que posteriormente se sustituyeron los textos por símbolos que permiten identificar los campos sin problema y reduciendo el espacio utilizado al mínimo (1 caracter).
De la misma manera, los botones de Personal, Laboral, Agregar, Editar, Eliminar y Guardar fueron agregados en una sola área común y ocupando el espacio de un solo caracter.
Los campos seleccionados para los contactos fueron los básicos de cualquier agenda de contactos, incluso el teléfono de casa estuvo a punto de no considerarse, sin embargo, dado que los campos son compartidos entre los datos de contactos personales y laborales, este campo era indispensable para el teléfono de oficina del contacto. Además, como se mencionó en alguna junta de definición de campos, no todo el personal, por increíble que parezca, tiene un número celular.

Para las citas, los usuarios comentaron que se sentían cómodos empleando Outlook por el tamaño de la interfaz y la gran facilidad que tienen de invitar a colegas a las reuniones. Sin embargo, muchos coincidieron que necesitaban una manera sencilla de agendar citas personales (o recordatorios, vistas de otra manera) de temas personales, pues no les gustaba que sus actividades personales aparecieran en el calendario laboral de *Outlook*®, ya que algunos diariamente imprimen esta información para traerla consigo. Entonces, la opción de las citas les pareció una excelente idea. La petición en este apartado fue que los recordatorios fueran similares a los de *Outlook*®. Así que, como se describió anteriormente en el apartado correspondiente, también se logró este objetivo, y es por eso que la interfaz de las citas es muy simple y no requirió tantos detalles.

La interfaz de las Notas fue la más simple, pues todos estuvieron de acuerdo desde el primer diseño y no requirió cambios durante el desarrollo de las versiones. Excepto al agregar la opción de búsqueda, que no se encontraba desde el inicio.
Esta opción surgió como petición de algunos usuarios, pues no tenían suficiente tiempo para buscar en los listados de contactos o notas la información y requerían una manera rápida, sencilla y eficiente para hacerlo.
Para los Reportes, de Uso y de Actividades, también fue necesario realizar algunas juntas y consensos pues algunos necesitaban más información y otros prácticamente la mínima.
Al analizar ambos casos, finalmente se decidió englobar la información en únicamente dos tipos de reportes, como quedó reflejado en las últimas versiones. El de uso es importante pues requerían conocer en qué programas invertían más su tiempo, para poder balancear mejor sus cargas de trabajo diario.
Aunque el de actividades fue el más solicitado por parte de los coordinadores y gerente, pues se necesitaba una manera rápida de determinar el número de tareas que tenían los empleados y de estas cuántas estaban pendientes y cuántas terminadas en un periodo corto.
Las gráficas, aunque sencillas, cumplen las solicitudes de los usuarios que simplemente querían ver de manera gráfica la información mostrada en números y que a simple vista no les dice mucho.
Estas gráficas proporcionan rápidamente un resumen de cómo están administrando el tiempo los empleados, que era el objetivo primario del programa.

Y finalmente para la configuración, se pidió que fuera lo más sencillo posible para no tener que cambiar tantas opciones y que el usuario tuviera que adaptarse al programa y no el programa al usuario, aunque como programadores sabemos que esto solo es cierto una vez que se ha entregado y no hay más tiempo de realizar actualizaciones. Sin embargo, por la naturaleza del programa y los usuarios fue necesario realizar versiones que finalmente tuvieron que adaptarse al usuario durante el desarrollo en la medida de lo posible.

El proceso que se siguió para las pruebas de las versiones fue desarrollar primeramente una versión, elegir un grupo de prueba o grupo piloto de 3 usuarios e instalarles la versión sin darles

muchos detalles del uso, para verificar lo intuitivo de la interfaz y sobre todo determinar qué tan cómodo se sentían con la misma. También se buscaba que el grupo encontrara errores que no había contemplado previamente.
Después de las primeras dos pruebas los resultados mostraban que se necesitaban cambios en todas las interfaces y se detectaron varios errores de validación que no habían sido contemplados en la programación, básicamente del tipo: si el botón A está seleccionado y se cambia la opción a D, entonces B debería estar activo y A solamente si D tiene información.

Algunos ejemplos de las primeras interfaces:


00
00
00
TUL info|74|Pendiente
Tema
Clas.
Tiempo (min)
Tema 4
TUL info
25
Tema 3
TUL info
2
Tema 2
TUL info
58
Tema 1
TUL info
74
P
T

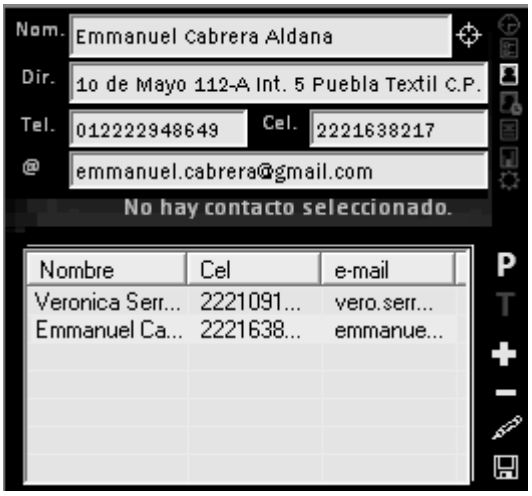
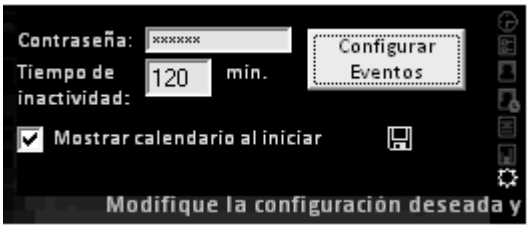

Nom.
Emmanuel Cabrera Aldana
Dir.
1o de Mayo 112-A Int. 5 Puebla Textil C.P.
Tel.
012222948649
Cel.
2221638217
@
emmanuel.cabrera@gmail.com
No hay contacto seleccionado.
Nombre
Cel
e-mail
Veronica Serr...
2221091...
vero.serr...
Emmanuel Ca...
2221638...
emmanue...
P
T


Gráfico de Uso
Estadística de Uso
Reporte de Actividades
Estadística de Actividades
áfico.


Contraseña:
******
Tiempo de inactividad:
120
min.
Configurar Eventos
Mostrar calendario al iniciar
Modifique la configuración deseada y

Durante el diplomado, hubo también observaciones realizadas por los profesores que fueron tomadas en cuenta, por ejemplo, la capacidad de mover la interfaz principal por la pantalla no era fácil de ver y tenía un problema, solo podía hacerse al hacer clic en un área vacía y arrastrarla. Esto se cambió a una flecha de cuatro puntas que aparece al momento de colocar el cursor sobre los números que indican la hora, con lo cual ahora es muy sencillo saber que la ventana puede moverse en la pantalla y desde donde debe hacerse.
Otra mejora fue la del botón de Apagado. Anteriormente bastaba con dar doble clic en cualquier área de la interfaz que estuviera vacía (de manera similar al problema mencionado arriba) pero no había manera de saberlo y peor aún, si por accidente se hacía doble clic en un área vacía, el programa simplemente desaparecía de la pantalla sin un aviso siquiera.
Para atacar este problema se colocó un pequeño botón de apagado que como puede verse en las imágenes se encontraba en otras versiones en la esquina superior izquierda del primer dígito de la hora, pero que por estética fue cambiado a la parte superior del botón que muestra / oculta el calendario y la marquesina.
Una observación más que sirvió para mejorar el funcionamiento del programa, fue mostrar los cambios en tiempo real.
La manera en la que se dibujaba el calendario (una sola vez al iniciar) obligaba al usuario que al hacer un cambio en la configuración o al agregar un evento (que no podía hacerse en un principio dando doble clic en el día propio del evento) tuviera que reiniciar el programa para visualizar los cambios.
Esto fue duramente criticado por los profesores pues obligaba al usuario a realizar una acción desagradable cada vez que agregara un evento y que además de tener que detener cronómetros y guardar cambios, quitaba tiempo y concentración en lo que estaba haciendo. Así que tuvo que rediseñarse toda la construcción del calendario por medio de un procedimiento que pudiera ser llamado en cualquier momento y que actualizara la información en tiempo real tanto de los eventos como de cualquier cambio en la configuración. Lo cual, posteriormente, facilitó otras tareas que involucraban al calendario, como poder desplazarse entre meses, mostrar la semana santa, etc.

Un usuario, preocupado por la planeación anticipada de sus vacaciones estaba muy interesado en que pudiera colocarse la información de la semana santa en el calendario, pues al tratarse de una fecha que no es fija y que no sabía cómo calcular, le frustraba el tener que esperar a que se publicara el calendario oficial de días laborables del año para hacer su planeación.
En juntas posteriores varios usuarios más apoyaron la petición y sugirieron que de ser posible agregarlo estarían muy agradecidos.
Esto involucró varios días de investigación al respecto, pues el cálculo de esta información no era simple. La semana santa está determinada por el calendario lunar. El domingo de pascua o de resurrección corresponde al domingo inmediatamente posterior a la primera luna llena tras el equinoccio de primavera y se debe calcular empleando la luna llena astronómica. Esto reduce el periodo en un rango del 22 de marzo al 25 de abril.
Esto quiere decir que para calcular la semana santa, tendríamos que calcular primeramente las fases lunares.
En las versiones que ya incluían este dato, primeramente se había colocado el código que calculaba las fases lunares (que como se mencionó previamente se obtuvo de un código *PHP* libre en *Internet*) y de aquí se intentaba calcular la semana santa. Pero hubo algunas fechas que no coincidían del todo, por lo que investigando más, apareció el algoritmo de *Butcher*, que daba un resultado preciso en un rango superior de fechas. Y fue el algoritmo que finalmente quedó en el código del programa.
Al tener ahora el cálculo de las fases lunares, fue sencillo agregar un cuadro de imagen en la esquina inferior derecha del calendario donde se pudiera mostrar la imagen de la fase lunar actual como en muchos calendarios se hace.
La adición de esta información agradó a los usuarios y sobre todo que pudiera mostrarse como evento los jueves y viernes santos de cada año para conocer, con la anticipación que ellos desearan, esta información.

Conclusiones

Después de todas las etapas involucradas en el proceso de la creación de este programa, se obtuvo una versión que cubría la mayor parte de los requisitos y peticiones importantes de los usuarios. Y digo “la mayor parte” porque a pesar de que el crear versiones y hacer pruebas es una herramienta bastante útil para un software a la medida, también es un arma de doble filo, pues realmente el usuario siempre va a querer más y más. Fue muy fácil observar que cuando se les presentaba una versión y solicitaban algo que no se tenía o hacía bien, para la siguiente versión se entusiasmaban al ver que ya lo hacía y ahora querían algo nuevo. Esta situación se repitió una y otra vez y aunque en ocasiones las peticiones eran un poco descabelladas, debo reconocer que algunas de estas ideas no se me hubieran ocurrido puesto que no las necesitaba.

La experiencia adquirida fue muy enriquecedora en varios aspectos, desde entender las peticiones de los usuarios y transformarlas en código funcional, hasta la sutileza de poder decir no a algunas de ellas y poner límites en cuanto al alcance del proyecto.
Por cuestiones de tiempo, el programa quedó en la versión que se ha descrito en este documento, pero tiene aún mucho potencial, pues varias herramientas, reportes, mejoras en general, pueden ser implementadas posteriormente para construir un programa robusto que brinde a los empleados del área y de otras áreas y empleos diversos la ayuda que pueda mejorar sustancialmente la administración de su tiempo.

A continuación transcribo algunos de los comentarios recibidos sobre el programa por los usuarios.

“Me encantó la interfaz, es moderna, al estilo de las aplicaciones de los celulares y me ayuda con mis actividades”
“Me ayudó a descubrir cosas sobre mí que desconocía. Como darme cuenta que paso mucho tiempo en el trabajo, y descuido mucho mi desarrollo profesional. Quizá lo sospechaba, pero nunca lo había medido.”
“Ya descubrí por qué no me alcanza el tiempo en mi trabajo. Es difícil reconocer que pasa uno mucho tiempo en Internet o haciendo otras actividades o simplemente con muchos tiempos muertos”
“Por fin tengo una manera de mostrarle a mi jefe la cantidad de actividades que realizo durante el día y el por qué no puedo terminar todas siempre a tiempo.”
“Muchos de nosotros tenemos demasiadas ocupaciones y muchas de ellas con mucha pérdida de tiempo. Principalmente nosotros que vamos frecuentemente a juntas, no tenemos tiempo de hacer nada más. Creo que no es sano tener tantas juntas y ahora podemos comprobarlo y solicitar algunos cambios”
“Como coordinadores nos hemos dado cuenta que las cargas de trabajo se encuentran bastante desequilibradas, y esto influye directamente en la productividad de los empleados. Esta herramienta nos ha sido útil de diferentes maneras, pero principalmente para reevaluar la manera en la que distribuimos las tareas y la manera en la que las ponderamos y evaluamos”
“Teníamos la idea de medir de alguna manera el tiempo real que algunos proyectos ocupaban de nuestra jornada laboral, sin embargo teníamos una herramienta que nos permitiera hacerlo. Cronometrar las tareas sonaba algo complicado y sobre todo muy controlador. Sin embargo, para fines administrativos y financieros, con este programa y el apoyo de todos los colaboradores involucrados en dichos proyectos, pudimos medir de una manera más eficiente y real el tiempo empleado en ellos y así calcular costos reales con anticipación”
“Pues aunque al principio si daba un poco de flojera estar recordando que se debía activar el temporizador y todo eso, con el uso
“Me gustó, y me ha servido. Quisiera que hiciera otras muchas cosas, que ojalá se incluyan en la próxima versión”

Las principales aportaciones de esta aplicación son:

- Permite el control laboral del trabajador que tiene como principal medio de trabajo a la computadora.
- Se puede estimar con mayor precisión el tiempo dedicado a cada tipo actividad. Esto es muy útil para negociar tiempos asignados a las tareas por parte del jefe.
- Se puede justificar el tiempo empleado en un periodo a los diferentes tipos de actividades.
- Se pueden tomar decisiones durante el desarrollo de una actividad y no esperar hasta que se consuma el tiempo asignado para justificar prórrogas o informar de adelantos.
- Si esta agenda se comparte con el jefe, puede ser un arma de dos filos en cuestión de control laboral.
- Permite tener una imagen clara de nuestro desempeño laboral y no la imagen que creemos en el desempeño profesional.
- Incrementa la productividad.
- Justifica los aumentos.
- La regla de funcionamiento adecuado es la honestidad.
- Aumenta la eficiencia de grupos de trabajo.

BIBLIOGRAFÍA.

1. http://office.microsoft.com/es-es/outlook-help/novedades-de-microsoft-outlook-2010-HA010354412.aspx
2. http://www.microsoft.com/project/es/es/project-professional-2010.aspx
3. Ingeniería de Software. Un enfoque práctico. Pressman, R. Quinta Edición, Mc Graw Hill 2002.
4. Ingeniería de Software. Sommerville, I. Séptima Edición. Addison Wesley 2005.
5. http://ldc.usb.ve/~mgoncalves/IS2/sd07/clase1.pdf
6. http://www.elprofesionaldelainformacion.com/contenidos/1999/abril/agentes_inteligentes_definicion_y_tipologia_los_agentes_de_informacion.html
7. http://msdn.microsoft.com/es-mx/library/windows/desktop/ms633521%28v=vs.85%29.aspx
8. http://msdn.microsoft.com/es-mx/library/windows/desktop/ms646261%28v=vs.85%29.aspx
9. http://msdn.microsoft.com/en-us/library/windows/desktop/ms645620%28v=vs.85%29.aspx
10. http://msdn.microsoft.com/en-us/library/windows/desktop/ms645618%28v=vs.85%29.aspx
11. http://www.experts-exchange.com/Programming/Languages/Scripting/PHP/Q_26472761.html