Trabajo Fin de Grado

CREACIÓN DE UNA APLICACIÓN INFORMÁTICA PARA LA PREPARACIÓN DE LAS ACTIVIDADES DE INSTRUCCIÓN EN MONTAÑA

Autor

CAC. D. Antonio Torró Laguillo

Directores

Director Académico: Dña. Myriam Cilla Hernández

Director Militar: Cap. D. Gonzalo Velasco Serrano

Centro Universitario de la Defensa – Academia General Militar

Zaragoza, 2017

Agradecimientos

Quisiera expresar mi gratitud a todos aquellos que han contribuido a la realización de este trabajo. En primer lugar quisiera agradecer a mi tío Vicente, la ayuda ofrecida con el código para generar la aplicación informática. Además, quisiera agradecer al BCZM “Pirineos” I/64 la ayuda ofrecida durante las prácticas y a la 1ª Cía. del citado Bón. la acogida en integración que me brindó durante las Prácticas Externas. También quisiera agradecer a Dña. Myriam Cilla Hernández la ayuda ofrecida a la hora de tutorizar el trabajo y al Cap. D. Gonzalo Velasco Serrano durante las Prácticas Externas.

.

Abstract

Implementation of a computer application for preparing the activities of mountain drill.

Nowadays, the Spanish army are improving their systems continously. It had been considered the necessity for digitize documentation of different types. There are a lot of information in the units to save and doing it by documents would be very uncomfortable task overall, when someone wanted to consult.

It has been considered that making a computer application which digitize all information about platoons, will help to have better control of the personnel documentation.

Cause of the unknowledge of the program code at first, it was expected that the application did not achieve outstanding results. However, the results obtained have been good because the program allows to have a control of the personnel that has participate in the recorded activities, as well as offering the option to assign a mark for each one of them.

The application has been called Sarrio because of the connection of this animal with the mountain.

The program has the capacity to register different users (Platoon Commanders), which, once connected, can make different personnel and activity records.

After this, the records are stored in database tables that can only be accessed by the same user who made them. That is to say, each user has a series of tables that store the pertinent information to his subordinates and to the activities done by the Platoon.

The improvement of the program is continuous due to the fact that the program code can be improved by adding functions that today have not been reached.

In short, Sarrio offers the ability to the Platoon Commander to have a digitalized control over all the activities done by the unit as a group and individually.

Abreviaturas

AF ..... Actividad Formativa
AGM ..... Academia General Militar
BCZM ..... Batallón de Cazadores de Montaña
BOD ..... Boletín Oficial de Defensa
Bón. ..... Batallón
Cía. ..... Compañía
CUD ..... Centro Universitario de la Defensa
CUMAS ..... Cuadros de Mando
ET ..... Ejército de Tierra
FUTER ..... Fuerza Terrestre
I/A ..... Instrucción y Adiestramiento
IAE ..... Instrucción, Adiestramiento y Evaluación
JCISAT ..... Jefatura de los Sistemas de Información, Telecomunicaciones y Asistencia Técnica
MADOC ..... Mando de Adiestramiento y Doctrina
NT ..... Norma Técnica
PAP ..... Programa Anual de Preparación
Pn. ..... Pelotón
PU ..... Pequeña Unidad
RICZM ..... Regimiento de Infantería de Cazadores de Montaña
Sc ..... Sección
SIGINST ..... Sistema de Gestión de la Instrucción
TFG ..... Trabajo de Fin de Grado
Tte. ..... Teniente
Tcol. ..... Teniente Coronel
VB ..... Visual Basic

Índice

Índice de figuras

1. Introducción

La presente memoria recoge los resultados obtenidos durante la realización del Trabajo de Fin de Grado (TFG) titulado: “Creación de una alicación informática para la preparación de las actividades de instrucción en montaña”. Este trabajo está encuadrado en el plan de estudios de quinto curso del grado en Ingeniería de Organización Industrial que se imparte en el Centro Universitario de la Defensa (CUD) a los futuros oficiales del Ejército de Tierra (ET) en la Academia General Militar (AGM) de la ciudad de Zaragoza.

En resumen, el proyecto se basa en la creación de una aplicación informática que ayude al mando de sección de las unidades de montaña a realizar un registro del personal subordinado y de las actividades desarrolladas por los mismos, y en concreto, de secciones del BCZM “Pirineos” I/64. Debido a la diferencia que determina a este tipo de unidades frente a otras más convencionales en cuanto a la instrucción técnica se refiere, esta aplicación está orientada para el uso de los mandos de sección de las mismas, ya que contiene varios aspectos que deben contemplarse en ellas y no tienen fundamento en otras.

1.1. Sistema de Gestión de la Instrucción (SIGINST)

El Sistema de Gestión de la Instrucción (SIGINST) constituye la herramienta fundamental para registrar todas las vicisitudes de la Instrucción, Adiestramiento y Evaluación (IAE) del personal, permitiendo el planeamiento, conducción y control de actividades a nivel compañía, escuadrón, batería o unidad similar.[1] En la actualidad, el SIGINST ofrece al Mando la capacidad de realizar un estudio previo de las actividades a desarrollar para instruir a su Unidad. Las diversas ventajas que ofrece la aplicación vienen reflejadas en la Norma Técnica (NT) 06/15 (actualización 2017)[1], como pueden ser la agilización en la realización del Programa Anual de Preparación (PAP) o el seguimiento de las actividades realizadas por los miembros destinados en la Unidad, así como sus aptitudes reflejadas en un informe personal.

Esta aplicación fue desarrollada por el Mando de Adiestramiento y Doctrina (MADOC) entre 2009 y 2010. En un primer momento, la finalidad fue proporcionar un registro de las actividades de instrucción del personal de tropa. La Fuerza Terrestre (FUTER) creyó que era necesario un registro de las actividades realizadas por la tropa de todas las Compañías (Cía,s) del Ejército de Tierra (ET). Para ello, se creó una base de datos que daba a las Cía,s. la posibilidad de realizar dicho registro de una forma cómoda y estandarizada a nivel Ejército. La base de datos fue realizada por el actual Teniente Coronel (Tcol.) D. Eduardo Camacho Medina, a partir de la cual surgió el SIGINST.

Para la creación del SIGINST, se precisó de un informático que, bajo las indicaciones del Tcol., creó el código para convertir la base de datos en un programa ejecutable con acceso al

mismo desde la Intranet de Defensa. Desde la creación en torno a 2008 de la base de datos y hasta mediados de 2017, las actualizaciones y mejoras del SIGINST por parte del MADOC han sido continuas. Sin embargo, a mediados del presente año, esta responsabilidad ha recaído sobre la Jefatura de los Sistemas de Información, Telecomunicaciones y Asistencia Técnica (JCISAT). Además, tras tener la aplicación la capacidad de poder registrar las actividades de instrucción realizadas por la tropa, se dedujo que la aplicación podría ofrecer, aparte del informe de dichas actividades, herramientas destinadas al planeamiento, conducción y control en las pequeñas unidades (PU,s) del ET.[1]

A lo largo de estos años se ha promovido el uso del SIGINST por parte del MADOC, intentando concienciar de su utilidad a las unidades mediante diferentes medios, tales como conferencias. Sin embargo, existen ciertos inconvenientes que ponen trabas a su empleo. Por ejemplo, el hecho de que para hacer un informe acerca de una actividad, la misma tiene que haber sido programada con anterioridad, es decir, si se realiza una actividad de instrucción "*in situ*" durante unas maniobras y esta no ha sido programada y cargada previamente en el SIGINST, a la vuelta de las maniobras no se puede anotar el informe de la actividad. Además de esto, el formato de programas de instrucción que se pide a las Cía,s desde escalones superiores es antiguo, por eso mismo, los Jefes de Cía. optan por dejar de lado el formato ofrecido por el SIGINST y seguir utilizando el obligatorio . Esto puede deberse a que el SIGINST tiene un ámbito de uso a nivel PU, es decir, las Brigadas solicitan un formato de programas de instrucción antiguo ya que no utilizan el SIGINST. No obstante, existe una aplicación que sí que utilizan, el Sistema de Gestión del Adiestramiento (SIGAD). La intención en las futuras actualizaciones de la plataforma informática es enlazar el SIGAD con el SIGINST mediante las 3ª,s Sc,s de los Bón,s, con la finalidad de que se propicie el uso del sistema, convirtiéndolo en una herramienta de trabajo, y no simplemente en una opción alternativa para guardar información. La necesidad de realizar esta actualización, ya ha pasado a manos del JCISAT, actual órgano responsable del SIGINST.

Las mejoras que se han dado a lo largo del tiempo se han ido implementando atendiendo a las necesidades de las Unidades, y con el fin de facilitar el ejercicio del Mando a la hora de planear las actividades y realizar su registro una vez realizadas. Entre estas mejoras cabe destacar la incorporación al SIGINST de los diferentes manuales divididos en fichas que ayudan a los Jefes de Cía. a programar las actividades. En las Planas de Mando de los Bón,s de las diferentes Unidades, es la 3ª Sc. (Operaciones) la que se encarga de programar las actividades de instrucción de las Cía,s en su conjunto. Esto hizo que se considerara la necesidad de que a los miembros de las 3ª,s Sc,s se les introdujera al SIGINST como usuarios del sistema. Debido a los roles propios de los mandos jefes de Pelotón (Pn.), Sección (Sc.) y Cía., se contempló que también era necesario introducirlos como usuarios en el sistema a causa de las funciones que desempeñan junto al personal de tropa, por lo que el registro de las actividades también empezó a realizarse sobre los Cuadros de Mando (CUMAS) hasta nivel Cía. y además se les dio la

capacidad de consultar o modificar los datos e informes del personal bajo su mando. Por último, se atendió a la necesidad de introducir como usuarios del sistema a todos aquellos que realizasen una Actividad Formativa (AF) del ET. Por eso mismo, en la última actualización, según el Tcol. D. Eduardo Camacho Medina, se introdujeron las AF,s en los que se incluyen los tres niveles propios de las Unidades de Montaña. Estos nuevos niveles (Elemental, Básico y Avanzado) que aún no están reconocidos oficialmente ya que los manuales que los determinan están pendientes de aprobación, provienen de los antiguos niveles no oficiales (Cazador, Esquiador/Escalador y Guía, respectivamente) que solo estaban reconocidos en las Unidades de Montaña como tal. Una vez que se aprueben los manuales de los nuevos niveles y se publiquen en el Boletín Oficial de Defensa (BOD), pasarán a ser oficiales y tendrán un reconocimiento a nivel Ejército. Sin embargo, este proceso puede llevar consigo un largo periódo de tiempo. Finalmente, se destaca que la instrucción en las unidades de montaña tienen que tener un tiempo dedicado a las actividades técnicas específicas de montaña, y no solo pueden dedicarse a la instrucción y adiestramiento (I/A) propios de otras unidades. El SIGINST actualmente solo contempla actividades comunes de I/A.

En resumen, el SIGINST se puede considerar un medio para digitalizar toda la información concerniente a la Instrucción de las Cía,s, lo cual es una ventaja ya que ofrece la capacidad de almacenar una gran cantidad de información que puede consultarse de una forma rápida y cómoda, ya sea referente al personal subordinado o a las actividades programadas. Así mismo es una fuente que ofrece documentación para la realización de ejercicios ya sean generales o específicos de algunas Unidades. Su ámbito de uso de la aplicación se extiende desde el personal de tropa hasta el empleo de Tcol. para las unidades de la Fuerza. Esto quiere decir que las utilidades que ofrece la aplicación son propias de las PU,s, desde nivel Pn. hasta Bón . Finalemente, se destaca su capacidad de mejora y el contínuo desarrollo y evolución que sufre para adaptarse a las diferentes necesidades del ET. Sin embargo, mientras algunas de estas mejoras se implementan, y en especial, la inclusión en la aplicación de los tres niveles propios de las Unidades de Montaña en las AF y la inclusión de actividades técnicas específicas propias de este tipo de unidades del ET, tales como el Regimiento de Infantería de Cazadores de Montaña (RICZM) “Galicia” 64, o más concretamente, el BCZM “Pirineos” I/64, han querido desarrollar su propia versión del módulo del SIGINST referente a la gestión de actividades para poder llevar un registro de las actividades realizadas por el personal del batallón.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es la creación de una herramienta informática que ayude al mando de sección de las unidades de montaña a realizar un registro del personal subordinado y de las actividades desarrolladas por los mismos. De esta forma, las unidades de montaña tendrán su propia versión del SIGINST, adaptada a sus necesidades especiales, mientras el JCISAT implementa las mejoras que se han comentado anteriormente. Además, cabe

destacar, que la aplicación informática aportará una considerable reducción del tiempo empleado en el seguimiento de las diferentes actividades realizadas y del personal que las lleva a cabo, mejorando la gestión del mismo.

Los requisitos y objetivos que la aplicación a desarrollar debe cubrir, están basados fundamentalmente atendiendo a la propuesta realizada por el RICZM “Galicia” 64, impulsor de la idea de realizar esta aplicación informática para satisfacer las necesidades del Regimiento. Desde esta unidad, se definieron unas pautas para que la aplicación informática ofreciera al mando de Sc. diversas facilidades que harán que la fase de planeamiento de las actividades de montaña se realice de una forma estandarizada y eficiente. Entre los requisitos que la aplicación tiene que proporcionar al jefe de Sc. se puede destacar:

- Listado completo y detallado de las actividades posibles a realizar por el personal subordinado dentro de la unidad de montaña, como por ejemplo, prácticas de rescate con diferentes materiales, recorridos topográficos o actividades de esquí.
- Listado del equipo específico a portar para cada una de las posibles actividades a realizar en una unidad de montaña.
- Listado del personal bajo el mando del usuario registrado y diversas características del mismo, ya sean administrativas, relativas al servicio o a la instrucción de combate o instrucción técnica de montaña.
- Histórico de actividades realizadas por los miembros de la Sc. con anotaciones que otorguen una valoración que bareme la forma en cómo ha desarrollado la actividad cada uno de ellos. Además, se incluirá una valoración común que represente el nivel de la propia Unidad en su conjunto.

En esta aplicación se debe poder dar de alta a todos los usuarios que se desee, y además, es importante que se puedan registrar los datos de las actividades realizadas, aunque estas ya hayan tenido lugar. Debe ser una aplicación informática flexible y fácil de usar.

1.3. Metodología

Para llevar a cabo este TFG, se han realizado diferentes etapas. En primer lugar, se ha realizado una revisión bibliográfica y del estado del arte de los diferentes manuales del ET. Además, se han revisado los manuales e información sobre el SIGINST, debido a la similitud con la aplicación a desarrollar en cuanto al registro de actividades y listado de personal se refiere. Esto permitió conocer la situación actual del SIGINST y las diferentes características de la gestión de personal y actividades. En segundo lugar, se ha consulta a diferentes expertos de las unidades de montaña sobre los requerimientos del programa, sobre las diferentes actividades a incluir en el programa y las bases de datos necesarias para su correcto funcionamiento. Estos expertos son principalmente los mandos diplomados de la unidad que, durante la realización de las prácticas externas, facilitaron toda la información necesaria para el estudio de las necesidades para la mejora del mando y control de las Sc,s. En especial, cabe destacar que ña

información proporcionada por el Capitán D. Carlos Manuel García Galindo y el Capitán D. Joaquín Pelegrín Gayán, ambos pertenecientes a la 3ª Sc. del BCZM “Pirineos” I/64, ha facilitado el enfoque para la realización de la aplicación informática a desarrollar. Durante esta fase, por tanto, se definieron los requisitos del programa a desarrollar. La tercera fase consistión en la elección del lenguaje de programación a utilizar. Se valoraron diferentes programas, pero finalmente se optó por el uso de Visual Basic.Además, se consultaron varios manuales y tutoriales relacionados con este lenguaje de programación, ya que el autor de este TFG nunca había programado ningún código en Visual Basic. Posteriomente, se desarrolló la propia aplicación, iterando siempre con la opinión de los expertos consultados. Finalmente, la última fase corresponde a la elaboración de la presente memoria, donde se ha dejado plasmado el trabajo realizado, y se han definido los trabajos futuros a realizar.

2. Lenguaje y herramientas de programación

2.1. Lenguaje de programación: Visual Basic

Tras valorar diferentes lenguajes de programación como por ejemplo Fortran, C++, Visual Basic o Pascal, se decidió que el programa se iba a desarrollar con el lenguaje de programación Visual Basic.net. El autor del presente TFG desconocía este código de programación, aunque si que tenía nociones de programación en Pascal gracias a las diversas asignaturas durante el grado. Sin embargo, a pesar de no tener experiencia con este lenguaje de programación, se decidió usarlo ya que el programa Visual Basic 2010 Express, que utiliza el lenguaje de programación Visual Basic .net., ofrece la capacidad de generar formularios propios de Windows, ofreciendo al usuario una interfaz cuyo uso se puede considerar intuitivo en el caso de que esté familiarizado con los sistemas operativos de Windows. Esto convertirá a la aplicación en una herramienta intuitiva, de fácil uso y muy visual.

Además, a nivel del desarrollador del programa, Visual Basic ofrece ventajas frente a otros códigos contemplados, como por ejemplo, la facilidad para recopilar información sobre el uso del código o, como en el caso concreto de este proyecto, la capacidad de hacer conexiones a bases de datos de Microsoft Office. Por ejemplo, en la Figura 1 puede verse como la pantalla “Objetivos del Proyecto” incluye el botón “Salir”, el cual ha podido generarse sin necesidad de tener ningún conocimiento en el lenguaje de programación, simplemente utilizando una herramienta propia del programa que genera el botón (Figura 2). No obstante, en la Figura 3 puede verse que se precisa del uso del lenguaje de programación para hacer que al pulsar el botón “Salir” se cierre la ventana “Objetivos del Proyecto”.

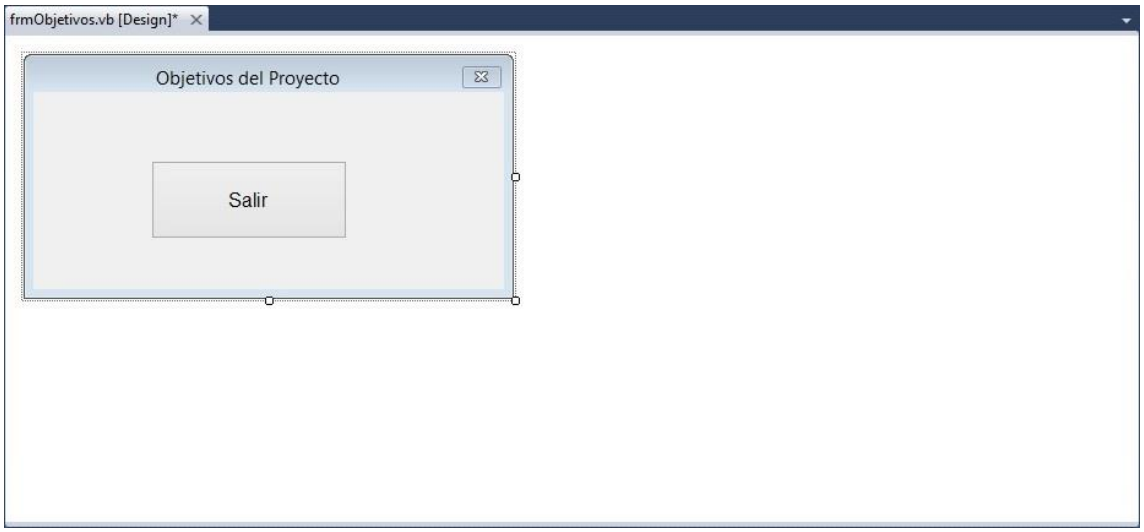


Figura 1. Vista de diseño de un Formulario de Windows.

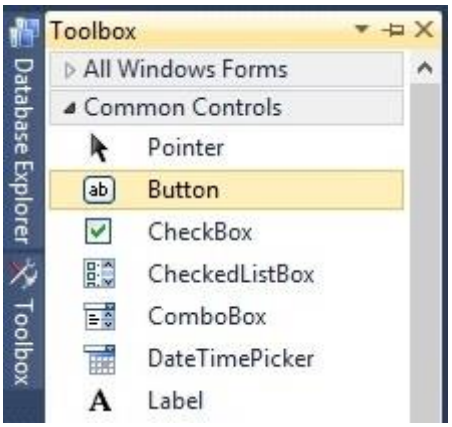


Figura 2. Parte de la caja de herramientas ("Toolbox").

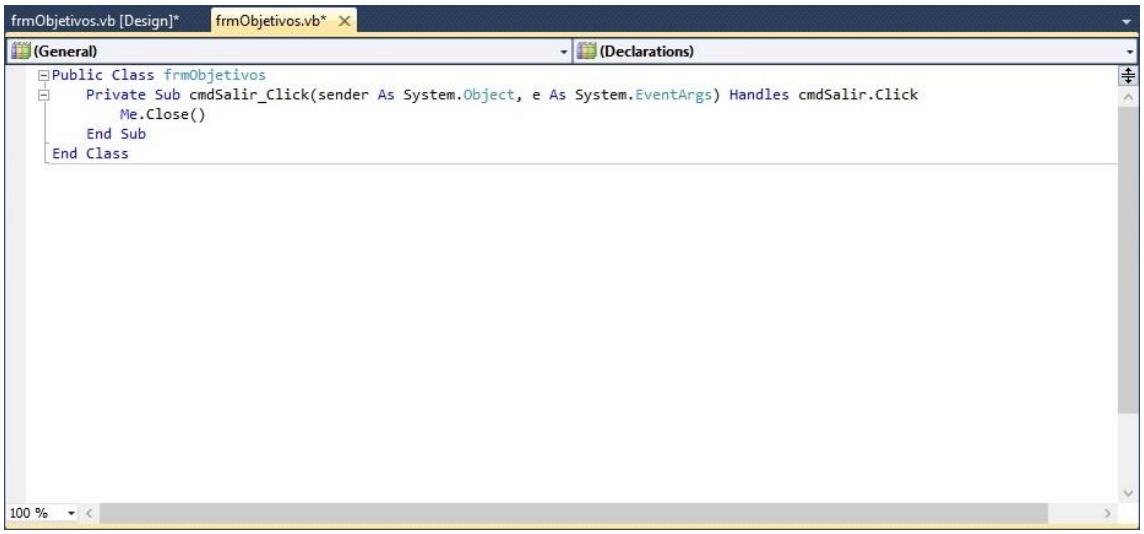


Figura 3. Vista de código de un Formulario de Windows.

2.2. Aspectos básicos de la aplicación

2.2.1. Acceso a la aplicación

En primer lugar, para poder acceder al programa es necesario haberse registrado previamente como “Nuevo Usuario” Una vez se ha realizado el registro, el usuario puede rellenar los datos personales que el programa le solicita para poder dar de alta al usuario. En caso de no introducir estos datos o no hacerlo correctamente, , no podrá acceder al programa y el programa le avisará de que no ha realizado correctamente el registro , y además, cada vez que un usuario registrado pero no dado de alta correctamente intente entrar al programa, será redirigido a la pantalla propia de rellenar los datos personales. Si el usuario se registra y da de alta correctamente, cada vez que quiera entrar de nuevo al programa, simplemente tendrá que rellenar su nombre (“Login”) y su contraseña.

Inicialmente no se contempló como objetivo del proyecto incluir un registro en el acceso a la aplicación. Sin embargo, finalmente se decidió realizar una base de datos para que quedaran registrados los datos de los diferentes usuarios que fueran a utilizar la aplicación. Esta medida facilita que si la aplicación se utiliza en un mismo equipo informático por más de un usuario, al acceder a la misma con sus datos, todos ellos sean capaces de ver únicamente lo que compete a su Sc., ya que, de otra forma, si varios usuarios introdujeran datos en las mismas tablas de las mismas bases de datos, crearía listados de personal y de actividades de todos los usuarios mezclados entre sí.

2.2.2. Actividades de instrucción en las unidades de montaña

Como ya se ha mencionado anteriormente, la instrucción en las unidades de montaña tiene que tener un tiempo dedicado a las actividades técnicas específicas de montaña, y no solo puede dedicarse a la I/A propios de otras unidades Por eso mismo, la aplicación será de uso exclusivo para este tipo de unidades. Aparte de las diferentes actividades que sí que se realizan en montaña y no tienen sentido en otras unidades, existen otras actividades comunes a otras unidades, como por ejemplo las marchas, que se registran de forma diferente en las unidades de montaña. Los cuadros de marcha propios de marchas en montaña, donde se muestra la dificultad o el desnivel acumulado, no son muy útiles para otras unidades que no realizan marchas por montaña, cuyo registro es únicamente la distancia recorrida.

Los cuadros de marcha que se representarán mediante la aplicación informática a desarrollar se basarán en los mismos que ya se realizan en las unidades mediante tablas de datos (ver Figura 4). Estos cuadros consisten básicamente en una tabla Excel que contiene fórmulas que calculan la distancia recorrida, el desnivel acumulado y el tiempo estimado de realización de la marcha, teniendo en cuenta diferentes puntos del itinerario, introducidos por el usuario manualmente en la propia tabla de datos. Además, permiten introducir el ritmo estimado que se pretende llevar a lo largo de la marcha dependiendo de si el desnivel es positivo, negativo

o de si el desplazamiento es en llano. La Figura 4 muestra un ejemplo de un cuadro de marcha proporcionado por el Cap. D. Joraquín Pelegrín Gayan donde se pueden observar cada uno de los elementos característicos de un cuadro de marcha y la información que este aporta.

FECHA: | ITINERARIO: Candanchu / Ibon de Estanes

	TRAMOS (sobreescribir el nombre en la columna izda sobre el pto correspondiente)		PTO INICIAL X	Y	Z	ORIENT	RUMBO	DISTANCIA REDUCIDA	DISTANCIA ACUMULADA	DIF NIVEL	PENDIENTE	TIEMPO TRAMOS	TIEMPO ACUMULAD
1	CAFETERIA GRANDE	PTO TORTIELLAS	701.503	4.740.280	1.550	205º	207º	1.703	1.703	450	26 %	1h 30' 00"	1h 30' 00"
2	PTO TORTIELLAS	CZE DE CAMINOS	700.785	4.738.736	2.000	215º	217º	412	2.115	0	0 %	0h 06' 10"	1h 36' 10"
3	CZE DE CAMINOS	COLLADO	700.551	4.738.397	2.000	242º	244º	1.596	3.711	500	31 %	1h 40' 00"	3h 16' 10"
4	COLLADO	PICO ASPE	699.146	4.737.640	2.500	84º	86º	124	3.834	140	113 %	0h 28' 00"	3h 44' 10"
5	PICO ASPE	COLLADO	699.269	4.737.653	2.640	264º	266º	124	3.958	-140	-113 %	0h 14' 00"	3h 58' 10"
6	COLLADO	CZE DE CAMINOS	699.146	4.737.640	2.500	62º	64º	1.596	5.554	-500	-31 %	0h 50' 00"	4h 48' 10"
7	CZE DE CAMINOS	PTO TORTIELLAS	700.551	4.738.397	2.000	35º	37º	412	5.966	0	0 %	0h 06' 10"	4h 54' 20"
8	PTO TORTIELLAS	CAFETERIA GRANDE	700.785	4.738.736	2.000	25º	27º	1.703	7.669	-450	-26 %	0h 45' 00"	5h 39' 20"
9	CAFETERIA GRANDE		701.503	4.740.280	1.550								
10													
11													

VEL ASCENSO (m/hora)	VEL DESCENS (m/hora)	VEL LLANO (km/hora)
300	600	4
PTE CRITICA		
5%		

TIEMPO CALC	5h 39' 20"
10% ALTOS	0h 33' 56"
10% IMPREVIST	0h 37' 20"

DIST RED TOT	7,67 km
ASCENS ACUL	1.090 m
DESCENS ACUL	-1.090 m
TPO TOTAL	6h 50' 36"

Fuente: Cap. D. Joaquín Pelegrín Gayán

Figura 4. Ejemplo de cuadro de marcha.

Además, la presencia de ambientación táctica, o no, en el ejercicio, es un hándicap que repercute en gran medida al mismo, ya que su presencia hace que varíe el desarrollo del mismo. Es decir, varía considerablemente una triple jornada de vida y movimiento en la que se desarrollan tres marchas frente a una triple jornada de un tema de combate en el que la distancia recorrida pueda ser la misma pero que se desarrolle de una forma mucho más lenta ya que se puede contemplar la presencia de tropas enemigas y la unidad pueda verse envuelta en un combate de encuentro (simulado según la ambientación del ejercicio).

2.2.3. Material y Equipo

El equipo a portar variará dependiendo del tipo de actividad que se vaya a realizar y por tanto, vendrá sujeto a las necesidades especiales de las unidades de montaña. El equipo de este tipo de unidades incluye elementos que no tienen sentido en otras, como por ejemplo, el uso de esquís que solo está ligado a actividades de montaña.

Debido a la variedad de actividades, ya sean técnicas de montaña o propias de instrucción de combate, el equipo marcado para cada una de ellas varía dependiendo de lo que se precise para su realización. En primer lugar, es importante saber el material y equipo a portar para poder realizar la logística necesaria para preparar la actividad. Pero, en segundo lugar, es importante tener en cuenta que el material y el equipo hace que varíe el peso que debe portar cada miembro de la Sc. y por tanto, haya que tenerlo en cuenta a la hora de programar aspectos

de la actividad a realizar, como por ejemplo es importante considerarlo a la hora de decidir el ritmo con que se pretende realizar una marcha.

2.2.4. Listado de personal e histórico de actividades

El listado de personal a desarrollar reflejará los valores individualizados de cada uno de los componentes de la Sc., es decir, mostrará aspectos como por ejemplo la edad, el sexo, la constitución física, la antigüedad de empleo, la antigüedad en la Unidad o el nivel técnico de montaña alcanzado.

Por otro lado, el histórico de actividades realizadas consistirá en un cuadro que refleje todas aquellas actividades que ha hecho la Sc. en su conjunto y las que ha realizado cada uno de los miembros que la componen, de forma que se tenga constancia de quién se ha ausentado en alguna jornada de instrucción o períodos de maniobras y de quién ha realizado las actividades. Además, mostrará la correspondiente valoración que el Teniente (Tte.) Jefe de Sc. estime que debe darle.

2.2.5. Bases de Datos utilizadas por la aplicación

El programa utiliza Bases de Datos Access que contienen diferentes tablas procedentes de la propia instalación del ejecutable. Hay algunas comunes como el peso del diferente tipo de material o la que almacena los datos de los diferentes usuarios registrados y otras que son propias de cada usuario. Las Bases de Datos han sido creadas por el autor del TFG y se generarán al ejecutar el instalador de la aplicación.

Respecto a las tablas utilizadas por la aplicación, se puede diferenciar tres tipos numeradas a continuación:

1. - Tablas comunes para todos los usuarios: En la **¡Error! No se encuentra el origen e la referencia.** puede verse que hay dos tablas creadas en la Base de Datos, las cuales, son comunes para todos los usuarios. Por ejemplo, se puede observar que en la tabla “datos” figuran una serie de campos de necesario cumplimiento para acceder a la aplicación. Tras el registro como nuevo usuario, el programa solicitará que se introduzcan los datos que pueden verse en la Figura 5 y en la Figura 6, a excepción de “Usuario”, que se habrá elegido con anterioridad.

datos

Usuario	DNI	Sexo	Empleo	Nombre	Apellido1
atorlag	72151852R	Hombre	Alférez	Antonio	Torró
ZZZZZZ	99999999Z	Mujer	Teniente	ZZZ	ZZZ

Figura 5. Tabla "datos" de la Base de Datos "tablas_comunes". (Parte 1)

Apellido2	Unidad	Compañía	FechaNac	AntEje	AntEmp
Laguillo	AGM	51	17/06/1991	22/08/2011	01/03/2016
ZZZ	ZZZ	1ª	27/10/1990	20/08/2009	12/07/2015

Figura 6. Tabla "datos" de la Base de Datos "tablas_comunes". (Parte 2)

En la tabla “Usuarios”, figura el listado de los usuarios registrados y sus respectivas contraseñas. Como puede verse en la Figura 7, el campo contraseña tiene de máscara de entrada el carácter *, lo cual hace que impida a terceros el poder leerla.

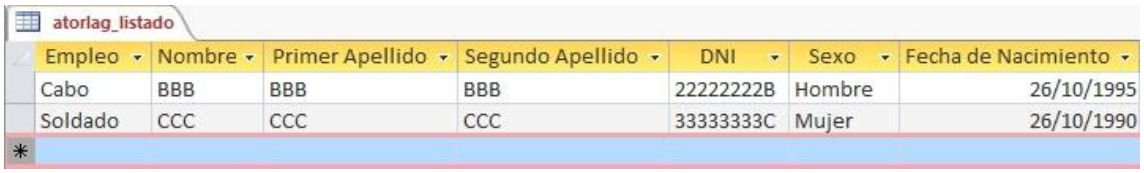
usuarios

Usuario	Contraseña
atorlag	**********
ZZZZZZ	******

Figura 7. Tabla de usuarios registrados y contraseñas de la Base de Datos "tablas_comunes".

2.- Tablas propias de cada usuario: Aparte de las tablas comunes a todos los usuarios, hay otras que son específicas de cada uno de ellos, como son las del listado del personal y las del histórico de actividades. Para que la Base de Datos que contiene estas últimas pueda guardar todas las tablas dependiendo del usuario que utilice la aplicación, sin que se generen problemas de que a varios de ellos se les asigne una misma, se ha añadido al nombre las tablas el del usuario, es decir, para un listado del personal o un histórico de las actividades realizadas, las tablas son calificadas como “’usuario’_listado” y “’usuario’_historico”, y ya que no puede haber más de un usuario con la misma cadena de caracteres de acceso (“*Login*”), no se pueden dar coincidencias en los nombres de las tablas.

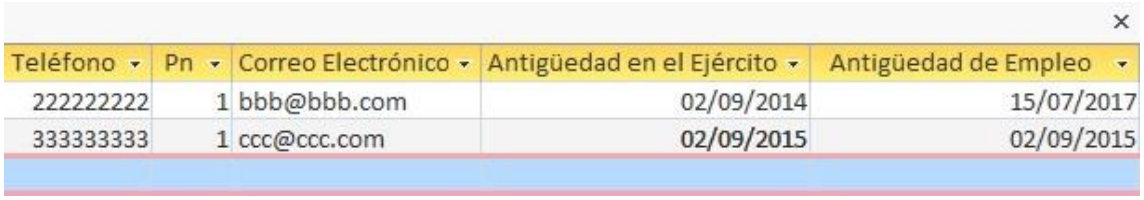
Las tablas de listado de personal (ver Figura 8 y Figura 9) contienen diferentes campos con la finalidad de generar una serie de resultados en la aplicación. Los registros en las tablas han de generarse a través de la aplicación. Así mismo, también podrán borrarse o modificarse. Debido a que dos personas no pueden tener el mismo DNI, el campo con ese nombre ha sido el elegido para ser la clave principal de la tabla, no pudiendo realizarse dos registros para un mismo DNI. Aparte de los campos presentes en Figura 8 y Figura 9, hay dos más: “ID” y “Check”. Estos campos facilitan la realización de consultas SQL por el programa, facilitando su desarrollo.



atorlag_listado

Empleo	Nombre	Primer Apellido	Segundo Apellido	DNI	Sexo	Fecha de Nacimiento
Cabo	BBB	BBB	BBB	22222222B	Hombre	26/10/1995
Soldado	CCC	CCC	CCC	33333333C	Mujer	26/10/1990

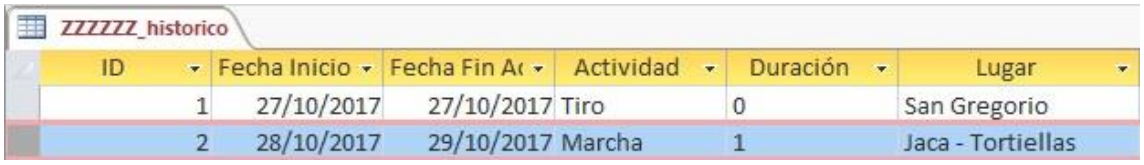
Figura 8. Ejemplo tabla de registro de personal de la Base de Datos "tablas_individualizadas". (Parte 1)



Teléfono	Pn	Correo Electrónico	Antigüedad en el Ejército	Antigüedad de Empleo
222222222	1	bbb@bbb.com	02/09/2014	15/07/2017
333333333	1	ccc@ccc.com	02/09/2015	02/09/2015

Figura 9. Ejemplo tabla de registro de personal de la Base de Datos "tablas_individualizadas". (Parte 2)

Las tablas del histórico de actividades sirven para registrar las actividades realizadas. El registro puede realizarse tanto antes de llevarlas a cabo, para programarlas, como después, en el caso de que la actividad se hubiera realizado sin haberla planeado previamente. Los datos que quedan registrados muestran las fechas de inicio y finalización de las actividades, su duración y el lugar donde fueron realizadas como se muestra en la Figura 10.



ZZZZZZ_historico

ID	Fecha Inicio	Fecha Fin A	Actividad	Duración	Lugar
1	27/10/2017	27/10/2017	Tiro	0	San Gregorio
2	28/10/2017	29/10/2017	Marcha	1	Jaca - Tortiellas

Figura 10. Ejemplo tabla de registro de las actividades de la Base de Datos "tablas_individualizadas".

3.- Tablas propias para cada uno de los subordinados: El programa debe dar la capacidad de obtener un informe de cada individuo que está registrado en "'usuario'_listado". El informe consiste básicamente en un registro de las actividades realizadas por el individuo en cuestión y la posibilidad de evaluar dichas actividades de forma individual. Para que el programa sea capaz de almacenar la información individualizada por cada actividad realizada, se ha decidido que cada nuevo registro en la tabla "'usuario'_listado" cree una tabla con su DNI en el nombre de la misma, de la forma "'DNI'_informe". De esta forma, se evitan posibles coincidencias en el registro de actividades, ya que la cadena de caracteres DNI no puede ser igual para dos individuos y a cada uno le corresponde una diferente. La Figura 11 muestra un ejemplo de una tabla propia para un subordinado.

2.2.6. Ventanas emergentes de aviso

Visual Basic 2010 Express ofrece una gran variedad de posibilidades a la hora de programar y, como es el caso, se ha enlazado la aplicación a desarrollar con bases de datos Access. Esto hace que se puedan generar varios errores ya no solo de compilación del ejecutable, sino una vez ejecutado el programa, con consultas, conexiones o modificaciones en

las tablas de las bases de datos, debido a errores de sintaxis de SQL que Visual Basic no detecta antes de compilar. Para detectar el problema, el lenguaje de programación ofrece la posibilidad de utilizar un comando (“Try”) para que, en caso de que no se llegue al resultado deseado, se abra una ventana emergente como excepción, de forma que puede localizarse más fácilmente el lugar donde el código está mal escrito.

3. Resultados

3.1. Esquema de la aplicación

La aplicación tiene una interfaz básica compuesta por una serie de ventanas. Las más importantes, que cabe destacar para facilitar el entendimiento del programa, son las mostradas en las Figuras**¡Error! No se encuentra el origen de la referencia.**,**¡Error! No se encuentra el igen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.** :

- Ventana Inicial
- Escritorio
- Nueva Actividad
- Listado de personal
- Histórico de actividades

La Se puede decir que a partir de la explicación de éstas y apoyándose con los esquemas plasmados en la Figura 11 y en la Figura 12, debería entenderse el funcionamiento de la aplicación., en primer lugar, a partir de la ventana inicial se realiza el “Login”. En el caso de que el usuario no esté registrado, se procede a realizar el registro del mismo en la base de datos tablas_comunes, mediante la opción “Nuevo Usuario”. Todo usuario debe rellenar una serie de datos personales para poder acceder a la aplicación, por lo que, en caso de no haberlo hecho tras realizar el registro como nuevo usuario, el programa lo detecta y al intentar realizar el “Login”, se abre una ventana emergente para introducir dichos datos. Si el usuario ya está registrado y ha rellenado sus datos personales, al realizar el “Login”, accede a la ventana llamada “Escritorio”, a partir de la cual se accede a las diferentes funcionalidades del programa.

Una vez en la ventana “Escritorio”, las funciones que ofrece el programa son “Nueva actividad”, “Listado de personal” e “Histórico de actividades”. También aparece la opción “Cerrar Sesión” que no se tiene en cuenta en la explicación actual, ya que simplemente se desconectaría al usuario del programa volviendo a la ventana inicial. Al seleccionar alguna de las otras tres funciones, se accede a una nueva ventana (diferente para cada una de las tres). En la ventana de “Nueva Actividad”, el usuario selecciona la actividad a realizar, introduce el lugar donde se realiza y la fecha y hora de inicio y finalización de la misma. Si la actividad dura menos de un día, la duración se indicará en horas. Tras elegir la actividad, el usuario elige el equipo y material que corresponda para el ejercicio seleccionado. Por último, se selecciona el personal que realizará la actividad. La actividad queda registrada en el “Histórico de Actividades” además de en el “Informe” de todo individuo que la haya realizado. En la ventana “Listado de personal” se ofrece

los datos de los individuos que han sido introducidos previamente por el usuario (subordinados). Con esta función se puede eliminar registros ya introducidos o añadir nuevos. Los registros introducidos o eliminados se guardarán aunque el usuario cierre sesión y vuelva posteriormente a iniciarla. Además, a partir del “Listado de personal”, se accede al informe individual de cada sujeto del propio listado. Finalmente, en la ventana “Histórico de actividades”, se muestra el listado de las actividades registradas a través de la función “Nueva Actividad”.

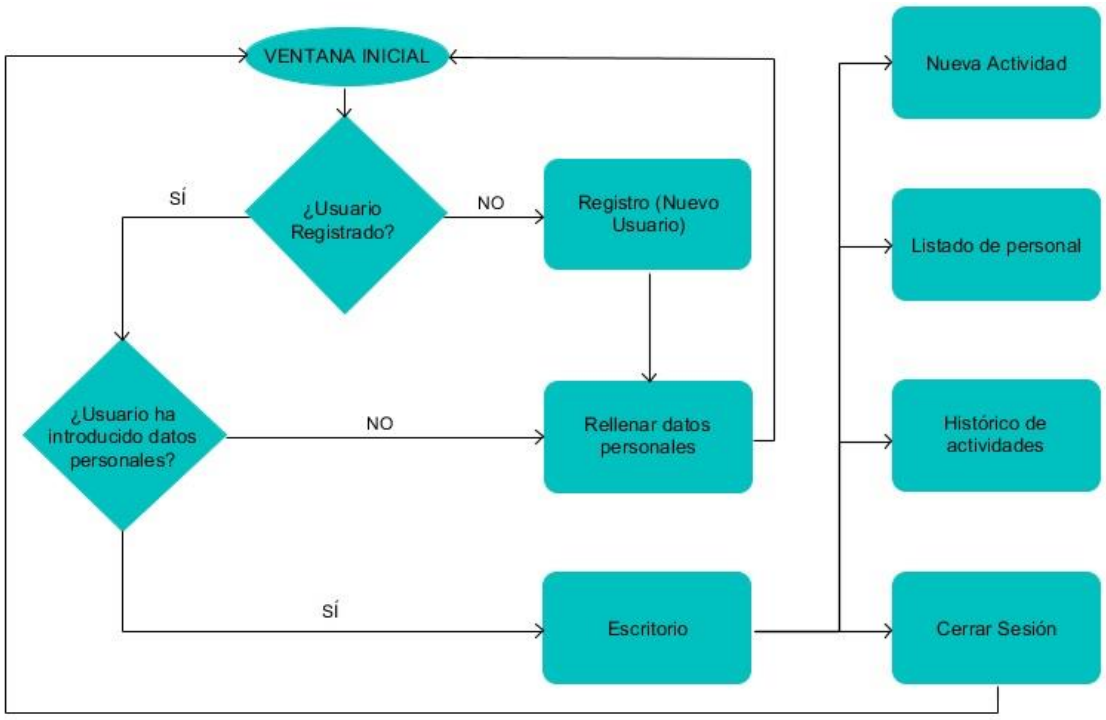


Figura 11. Esquema de la aplicación. (Parte 1)

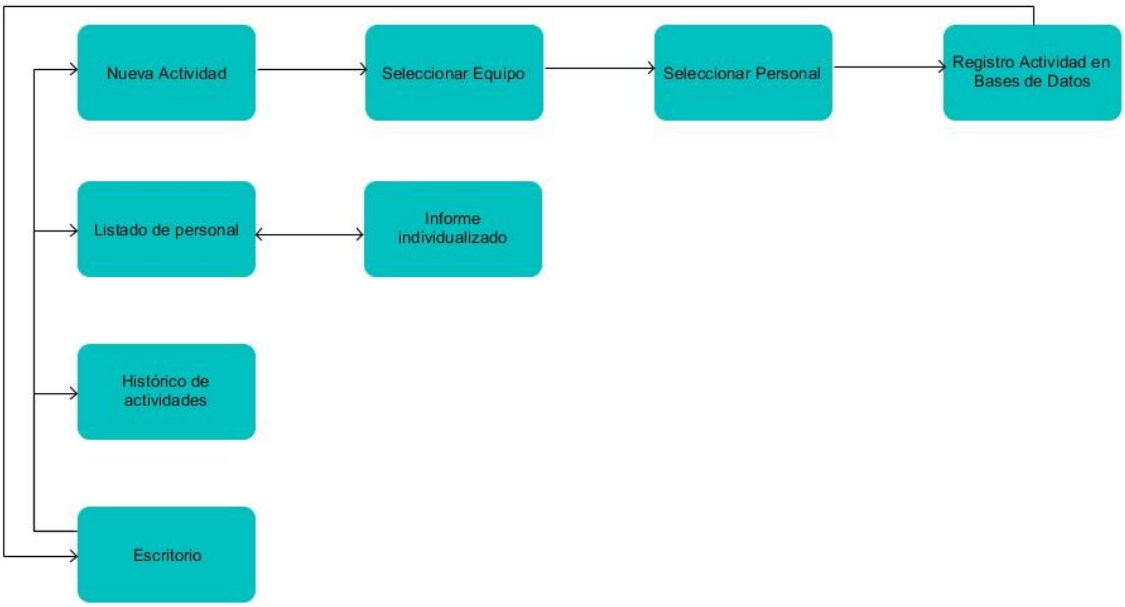


Figura 12. Esquema de la aplicación. (Parte 2)

El código en Visual Basic de la aplicación informática desarrollada se encuentra en el Anexo B.

3.2. Interfaz de la aplicación

En este apartado se muestra la interfaz del programa. Aunque ya se ha comentado anteriormente algunas de las funciones incluidas en este apartado, se ha considerado necesario, incluir una descripción de la interfaz gráfica. Debido al cuantioso número de figuras que se presentan en este apartado, se han plasmado todas ellas en el Anexo A, por lo que para toda referencia a una ilustración el lector deberá buscar en el citado anexo. Los resultados obtenidos de la implementación del código que se describen a continuación, siguen el esquema básico explicado en el apartado anterior.

Cabe mencionar que se ha programado una serie de ventanas emergentes y errores que saltan cuando se comete un error en la introducción de los datos (Figura 1 – Anexo A).

3.2.1. Ventana inicial "Sarrio – Preparación de Actividades en Montaña"

El usuario debe identificarse para acceder a la aplicación. Para esta aplicación el usuario deberá ser Jefe de Sc., es decir, por norma general un Tte. (Figura 2 – Anexo A). Como se ha comentado anteriormente, la opción "Nuevo Usuario" ofrece la posibilidad de registrarse en la base de datos siendo necesario únicamente la primera vez que se accede al programa (Figura 3 - Anexo A).

3.2.2. Ventana "Sarrio: Nuevo Usuario – Ficha de datos personales"

Una vez se ha realizado el registro, la aplicación solicita introducir una serie de datos personales, los cuales son de obligado cumplimiento para acceder al programa. (Figura 4 - Anexo A). En el caso de no introducir los datos, aunque el usuario quede registrado, cuando intente acceder a la aplicación, aparecerá una ventana emergente que indicará que no ha rellenado los datos pertinentes, (Figura 5 – Anexo A) por lo que se le redirigirá de nuevo a "Sarrio: Nuevo Usuario – Ficha de datos personales" para que los introduzca. Tras realizar el registro y rellenar los datos personales, el usuario puede acceder al programa.

3.2.3. Ventana "Sarrio: Preparación de Actividades en Montaña – Escritorio"

Esta ventana ofrece la posibilidad de programar una nueva actividad, ver el listado del personal subordinado o el histórico de actividades. La razón por la que se le ha nombrado "Histórico de Actividades" es porque las actividades realizadas quedan grabadas, no obstante, también se puede añadir registros de actividades futuras, por lo que aparte de un histórico de actividades es una programación de las mismas. Aparte de esto, una cuarta opción permite al usuario desconectarse del programa cerrando su sesión (Figura 6 – Anexo A).

3.2.4. Ventana “Sarrio: Preparación de Actividades en Montaña – Nueva Actividad”

Seleccionar la opción “Nueva Actividad” en el “Escritorio”, hace que el programa abra una nueva ventana (Figura 7 – Anexo A). El usuario podrá elegir el tipo de actividad a realizar seleccionando una opción de la lista desplegable (Figura 8 – Anexo A). En el campo “Lugar” se hará referencia a la zona geográfica donde se realice la actividad para que quede constancia en la base de datos y figure como descripción. A partir de la selección de la fecha de inicio y la fecha de finalización, el programa calcula la duración de la actividad al igual que en el caso de las horas, si fuera el caso en el que la actividad se iniciase y finalizase en el mismo día. Una vez insertados los datos en todos los campos, se abre una ventana que genera una tabla con el equipo y el material a seleccionar para la actividad (Figura 9 – Anexo A) y al pulsar “Aceptar” se genera un listado con el equipo y material seleccionado previamente, y un peso total aproximado por truncamiento (Figura 10 – Anexo A). Si se confirma el material seleccionado, una nueva ventana da la opción de seleccionar al personal registrado en “Listado de Personal” que realizará la actividad (Figura 11 – Anexo A) y posteriormente se generará un listado que habrá que confirmar. (Figura 12 – Anexo A). Finalmente, tras confirmar el personal que acudirá a la actividad, esta se almacena en el “Histórico de Actividades” y en el “Informe” del personal que la haya realizado.

3.2.5. Ventana “Sarrio: Preparación de Actividades en Montaña – Listado de Personal”

La opción del “Escritorio” llamada “Listado de personal” ofrece una tabla en la que figuran los datos de los diferentes subordinados registrados por el usuario (Figura 13 – Anexo A). El programa permite la modificación de los mismos, pudiendo añadir (Figura 14 – Anexo A) y eliminar (Figura 15 – Anexo A) miembros en la tabla. Cada Jefe de Sc. tiene una tabla propia en la base de datos con nombre del usuario, introducido al realizar el registro en la aplicación, como referencia. Seleccionar la opción “Informe”, hace que se abra una nueva ventana que muestra por un lado los datos del registro seleccionado y además su “Histórico de Actividades” personal, es decir, aquellas actividades que ha realizado o que va a realizar. En esta ventana, se permite la opción de agregar una calificación al individuo en cuestión para cada actividad registrada. (Figura 16 – Anexo A)

3.2.6. Ventana “Sarrio: Preparación de Actividades en Montaña – Histórico de Actividades”

El “Histórico de Actividades” muestra el historial del registro de actividades, ya hayan sido realizadas o se vayan a realizar. (Figura 17 – Anexo A)

4. Conclusiones y líneas futuras

4.1. Objetivos iniciales vs resultados obtenidos

El resultado final del proyecto ha sido el esperado en cuanto a funcionalidades de la aplicación se refiere. No obstante, la aplicación siempre es mejorable. En el futuro se puede implementar otras actividades, así como aumentar la información que se registra de las mismas. De todas formas, se ha mostrado la aplicación a los mandos de Sc. durante su desarrollo y la han valorado positivamente, ya que permite vincular las actividades y el listado del personal además de poder introducir registros de actividades ya realizadas.

Las diferentes trabas impuestas por las conexiones entre las herramientas de programación utilizadas y bases de datos u otros archivos, además de las avenidas por el uso del lenguaje SQL a través de Visual Basic, y teniendo en cuenta el desconocimiento del lenguaje de programación en un inicio, han hecho que la aplicación no se haya desarrollado tanto como podría haberlo hecho un tercero con conocimientos del lenguaje de programación.

4.2. Bases de datos y memoria requerida

Actualmente, una aplicación de esta índole, podría funcionar en la mayoría de los equipos informáticos sin que se generasen problemas. No obstante, para aplicaciones que requiriesen una mayor capacidad de memoria volátil, el modo en cómo se ha desarrollado el programa podría causar problemas. La base de datos llamada “tablas_individualizadas”, contiene una serie de tablas que han sido generadas en función del nombre del usuario o del DNI de los registros de personal. Esto hace que por cada usuario se creen dos tablas (“’usuario’_listado” y “’usuario’_historico”) y una más por cada registro de personal que haya hecho el usuario en el “Listado de personal” (“‘DNIsubordinado’_informe”). Es decir, para un número de usuarios con sus correspondientes subordinados registrados en el “Listado de Personal”, el número de tablas creadas sería equivalente a dos veces los usuarios más el número de subordinados registrados de cada uno de ellos. Esto se puede considerar un fallo de concepto, ya que mediante un procedimiento de relaciones entre tablas, podría haberse logrado que con, una sola tabla se generase en la aplicación el “Listado de personal” únicamente con sus subordinados para cada usuario. Si para una misma tabla se introducen los usuarios y los subordinados, a través de un diagrama de entidad-relación y con un campo denominado, por ejemplo, “Jefe”, se puede asociar a cada jefe varios subordinados, con una relación de cero o muchos a uno. (Véase Figura 13). Además, de este modo, se puede hacer que el jefe de Sc. tenga unos subordinados que a su vez tengan otros subordinados, es decir, se podría añadir a los jefes de Pn. y a los jefes de escuadra.

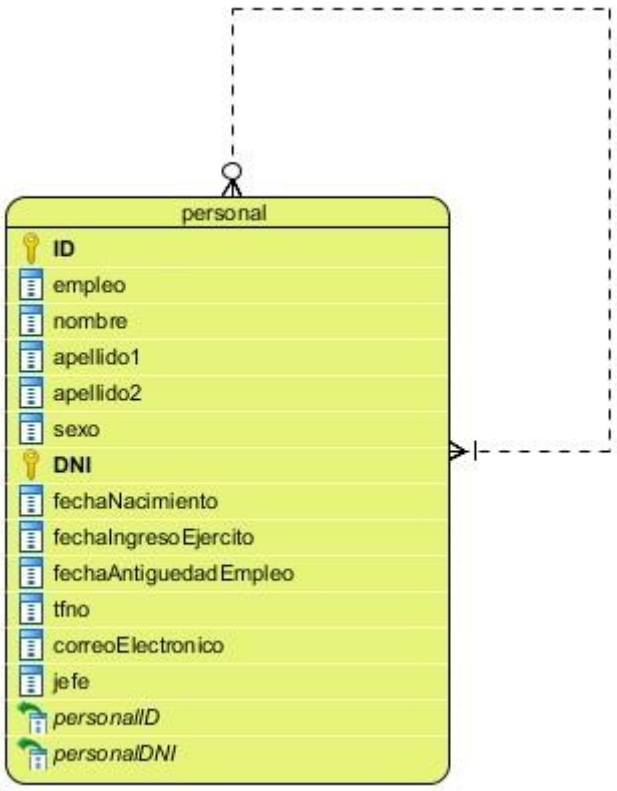


Figura 13. Ejemplo Entidad-Relación para el personal.

No obstante, si la aplicación se hubiera realizado de esta manera, en la tabla se hallarían todos los usuarios y subordinados registrados, y pese a que el programa mostrase a cada usuario únicamente su personal, en el caso de que se decidiese utilizar la base de datos sin utilizar la aplicación, los usuarios no dispondrían de una tabla con su personal, sino que dispondrían de una con todos los usuarios registrados y el personal subordinado. Se puede decir que haberlo hecho de esta forma, facilita al mando de sección extraer sus propias tablas de la base de datos en caso de que las necesitara utilizar para causas ajenas a la aplicación desarrollada, y como ya se ha dicho anteriormente, al ser una aplicación muy básica, la mayoría de los equipos informáticos con la memoria de la que disponen puede soportar un gran número de tablas para una misma base de datos sin que genere problemas de llenado de memoria.

4.3. Código VB .NET

El programa funciona correctamente y cumple todos los requisitos iniciales definidos por el RICZM “Galicia” 64. Sin embargo, a causa de los escasos conocimientos iniciales del código Visual Basic .NET, es posible que la extensión del mismo pudiera reducirse en gran medida, es decir, para unas funcionalidades iguales a las actuales, un tercero con conocimientos del lenguaje de programación y de su correcta aplicación, es posible que fuera capaz de sintetizar el código haciendo que el programa realizase las sentencias de una forma más eficiente, reduciendo el tamaño del ejecutable y errores de compatibilidad con las conexiones a las bases de datos y otros archivos.

4.4. Mejoras SQL

Con una mejora de las sentencias en lenguaje SQL utilizadas en el código de programación, se podría mejorar la aplicación. En este momento, existe la posibilidad de "hackear" la aplicación mediante lo que se conoce como "SQL-Injection", que permite con caracteres como "%", que una consulta analice todos los datos de una tabla y encuentre un valor buscado. Por ejemplo, para el caso de acceder a la aplicación mediante el "Usuario" y la "Contraseña", introduciendo los caracteres adecuados, a causa de la inyección SQL, un tercero puede acceder a la cuenta de un usuario sin conocer su contraseña.

4.5. Otros registros en las bases de datos a partir de la aplicación

Se puede incrementar los registros que se realizan en las bases de datos añadiendo campos en las mismas. Las posibilidades y opciones a añadir son infinitas. El autor de este proyecto se ha centrado en aquellos requisitos que se definieron inicialmente. Algunas de estas mejoras son, por ejemplo, para el "Informe Personal", se puede añadir un campo de texto largo en la tabla "'usuario'_listado" en el que se pueda introducir comentarios relativos al subordinado o bien en la tabla "'DNI'_informe", para no solo poner una calificación numérica sino además una valoración por escrito de cómo ha realizado el subordinado seleccionado cada actividad. Además, añadiendo un campo seleccionable como el que figura en el "Listado de Personal" (primera columna) para acceder al "Informe Personal" (Véase Figura 13 – Anexo A), se podría hacer que para actividad, se generase una plantilla estandarizada con diferentes aclaraciones acerca de la misma. Por ejemplo, en el caso de un ejercicio de tiro, existe la posibilidad de que, tras seleccionar el personal que lo va a realizar, se abra una nueva ventana en la que haya que rellenar la descripción de los ejercicios de tiro y la munición a consumir en cada ejercicio. Tras rellenar todos los campos, la actividad quedaría registrada en el "Histórico de Actividades" y en el "Informe Personal" de los subordinados que la hubieran realizado. A partir del recuadro seleccionable en dicho "Informe Personal", se accedería a una ventana en la que figuraría la descripción de los ejercicios de tiro y además existiría la posibilidad de añadir las puntuaciones obtenidas en cada uno de ellos.

Otro ejemplo sería para la función "Nueva Actividad", la aplicación genera un registro en el "Histórico de Actividades" y en el "Informe Personal", no obstante, en cuanto al equipo y material seleccionado, únicamente se genera un listado que no se almacena en ningún archivo, es decir, una vez que se registra una actividad no se puede volver a consultar el equipo y material seleccionado para la misma. Esto se debe a que se pretende emplear únicamente como referencia a lo que se va a utilizar en la actividad, pudiéndose hacer una captura de pantalla e imprimiendo el listado para informar a los subordinados a la hora de preparar el equipo y material pertinente. No obstante, existe la posibilidad de que dicho material quede registrado en una tabla de una base de datos. Sería un caso similar al "Informe Personal" en el que al seleccionar una actividad en el "Histórico de Actividades", se abriese una ventana con el listado del equipo y

material utilizado. Sin embargo, se puede considerar que dicho listado es útil para la preparación de la actividad y que ningún individuo se olvide algún elemento de la lista gracias a que pueda consultarla, pero no se percibe la necesidad de que quede un registro tras su realización.

Finalmente, al autor le gustaría añadir, que para la realización de las marchas no se ha conseguido realizar la conexión de la aplicación con la hoja de datos excel del cuadro de marcha tipo, imposibilitando que se mostraran los datos obtenidos a partir de las fórmulas de la hoja. En futuras versiones de la aplicación implementada, sería importante incluir esta información.

5. Bibliografía

5.1. Referencias bibliográficas de libros y documentos

[1] Ancos, Luis Miguel Blanco. *Programación en Visual Basic .NET*. Madrid: Grupo Eidos, 2002.

[2] DIEN; MADOC. «SISTEMA DE GESTIÓN DE LA INSTRUCCIÓN EN EL ET.» *NORMA TÉCNICA 06/15 (ACTUALIZACIÓN 2017)*. Granada: Ministerio de Defensa; Ejército de Tierra, Abril de 2017. 30.

[3] Halvorson, Michael. *Microsoft Visual Basic 2013*. Sebastopol: O'Reilly Media, 2013.

[4] Luna, Fernando O. *Visual Basic*. Buenos Aires: Fox Andina, 2011.

5.2. Referencias bibliográficas de sitios web

[5] Artavia, Lic. Pablo Jiménez. *https://www.youtube.com/watch?v=eyWFynDDD7g*. 26 de Julio de 2015. (Fecha de consulta: 26 de Octubre de 2017).

[6] Rojas, Juan Carlos. «Kubical ORG Software Studio.» 2010. *https://www.lawebdelprogramador.com/foros/Access/598498-Crear-tabla-de-Visual-Basic.html.* (Fecha de consulta: 13 de Octubre de 2017).

[7] *https://www.1keydata.com/es/sql/sql-create-table.php*. s.f. (Fecha de consulta: 13 de Octubre de 2017).

[8] *https://code.msdn.microsoft.com/windowsdesktop/access-2007-oledb-with-2fed4cc1*. 20 de Diciembre de 2013. (Fecha de consulta: 12 de Octubre de 2017).

[9] *http://www.elguille.info/NET/ADONET/ejemploSQL.htm*. 19 de Abril de 2003. (Fecha de consulta: 13 de Octubre de 2017).

[10] *https://msdn.microsoft.com/es-es/library/b5xbyt6f(v=vs.90).aspx*. s.f. (Fecha de consulta: 25 de Octubre de 2017).

[11] *https://msdn.microsoft.com/es-es/library/cc467034(v=vs.71).aspx*. s.f. (Fecha de consulta: 18 de Octubre de 2017).

[12] *https://msdn.microsoft.com/es-es/vba/access-vba/articles/calculate-elapsed-time*. s.f. (Fecha de consulta: 25 de Octubre de 2017).

[13] *https://msdn.microsoft.com/es-es/VBA/Access-VBA/articles/insert-update-and-delete-records-from-a-table-using-access-sql*. s.f. (Fecha de consulta: 25 de Octubre de 2017).

[14] *https://www.tutorialspoint.com/sql/sql-create-table.htm*. s.f. (Fecha de consulta: 18 de Octubre de 2017).

[15] *https://www.youtube.com/watch?time_continue=22&v=vhZnb5cws1c*. 06 de Enero de 2011. (Fecha de consulta: 12 de Octubre de 2017).

[16] *https://www.youtube.com/watch?v=0GTht3LTUU0*. 06 de Enero de 2011. (Fecha de consulta: 12 de Octubre de 2017).

[17] *https://www.youtube.com/watch?v=1EokvQzUgJE&t=633s*. 06 de Enero de 2011. (Fecha de consulta: 12 de Octubre de 2017).

[18] *https://www.youtube.com/watch?v=GpA3zS_IoMk*. 06 de Enero de 2011. (Fecha de consulta: 12 de Octubre de 2017).

[19] *https://www.youtube.com/watch?v=mi4cDGDPt7o*. 03 de Agosto de 2015. (Fecha de consulta: 26 de Octubre de 2017).

Anexo A

Capturas de pantalla de “Sarrio – Preparación de Actividades en Montaña”

Figura 1 - Anexo A. Ejemplo de mensaje de excepción.

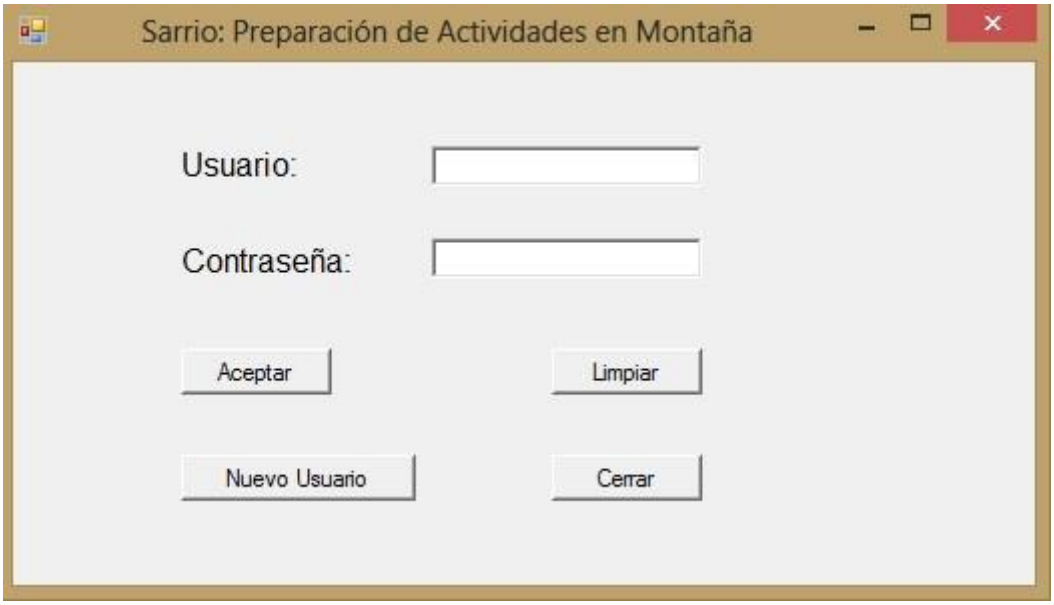


Figura 2 - Anexo A. Ventana inicial de la aplicación.

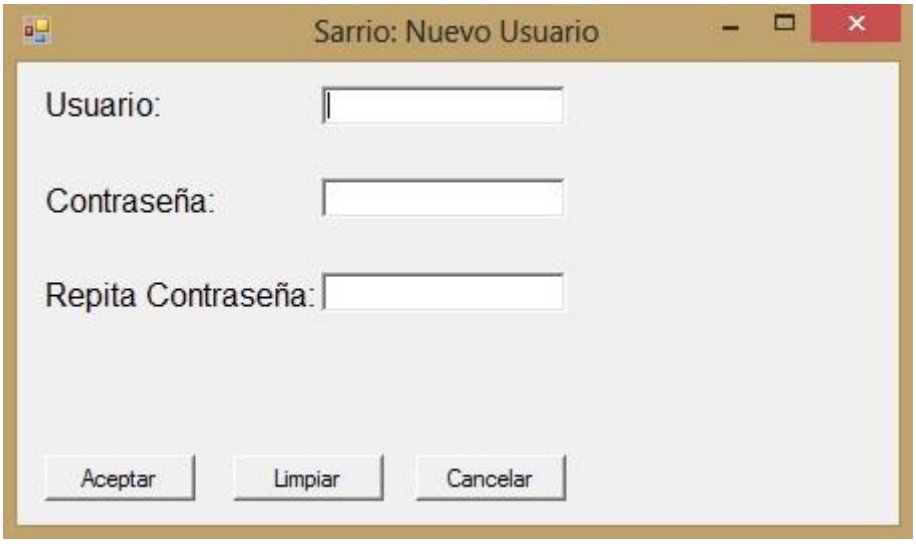


Figura 3 - Anexo A. Ventana de registro para nuevos usuarios.

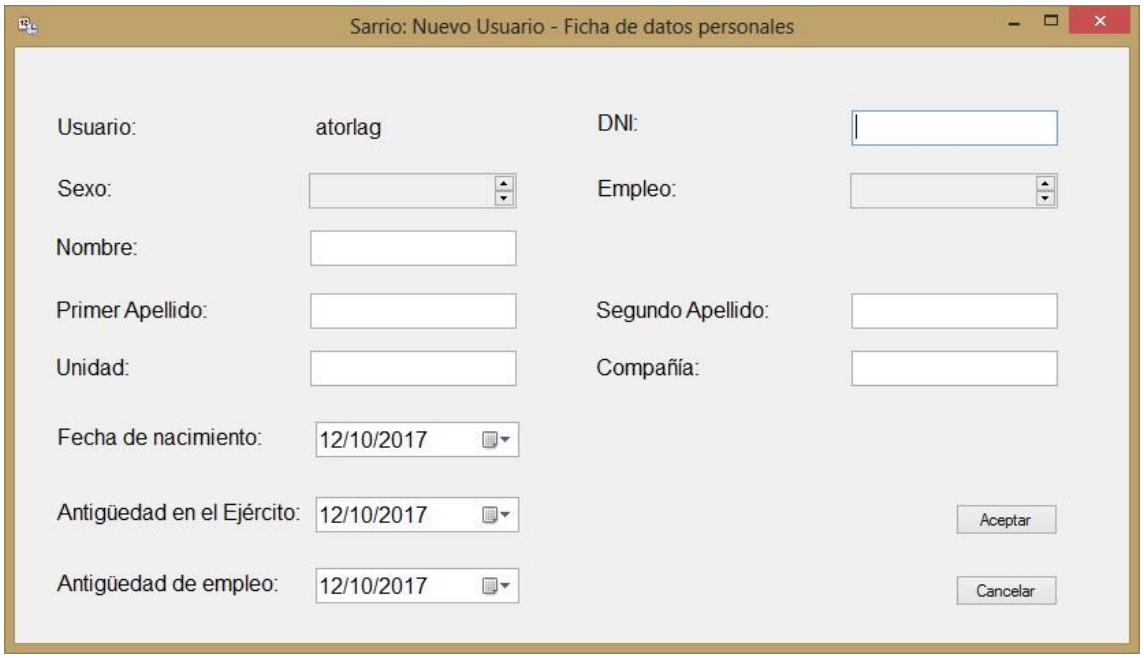


Figura 4 - Anexo A. Ventana para introducir los datos personales de un nuevo usuario.

Figura 5 - Anexo A. Ventana emergente por falta de datos personales.

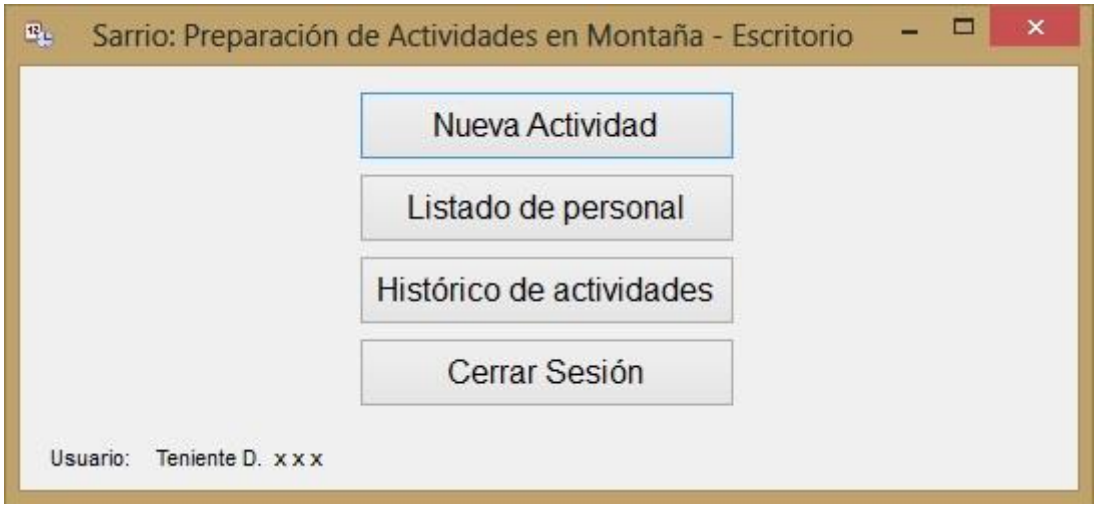


Figura 6 - Anexo A. Ventana "Escritorio".

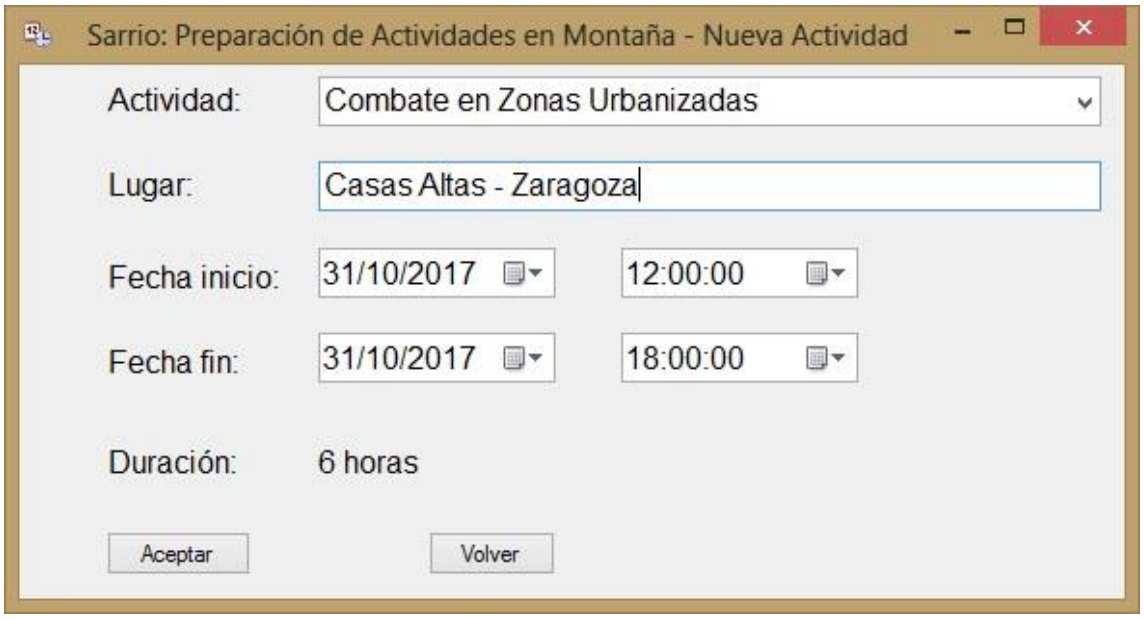


Figura 7 - Anexo A. Ventana "Nueva Actividad".

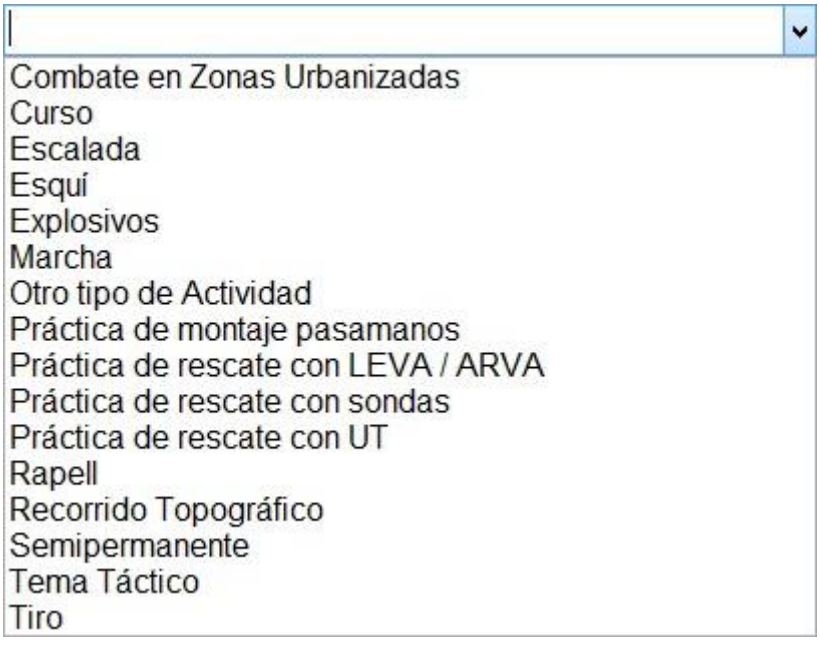


Figura 8 - Anexo A. Lista desplegable para elegir la actividad a realizar.

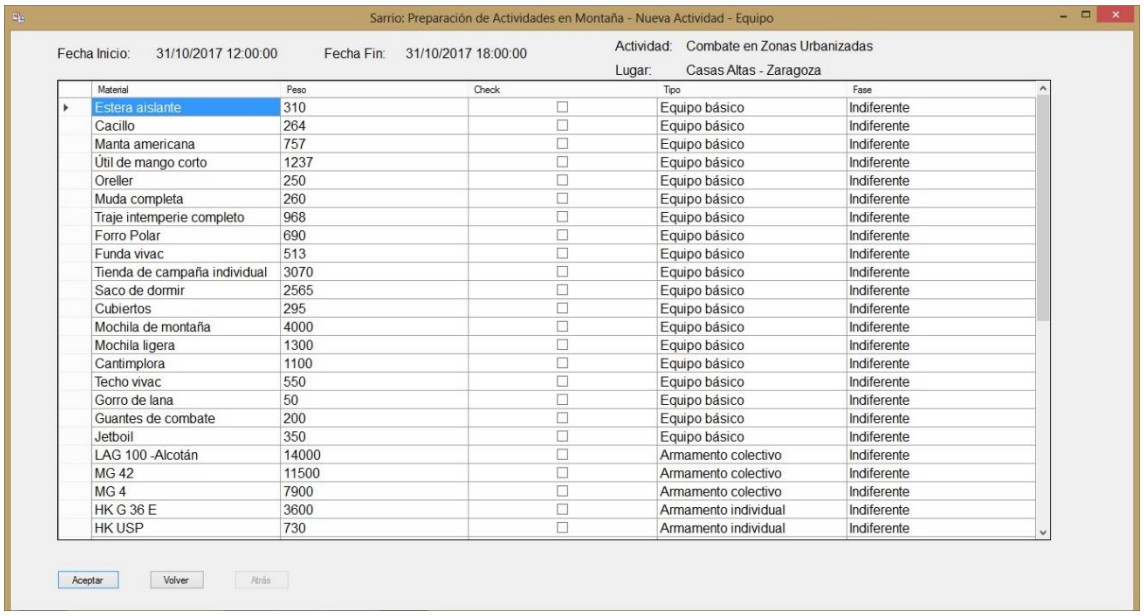


Figura 9 - Anexo A. Ventana de selección de equipo y material.

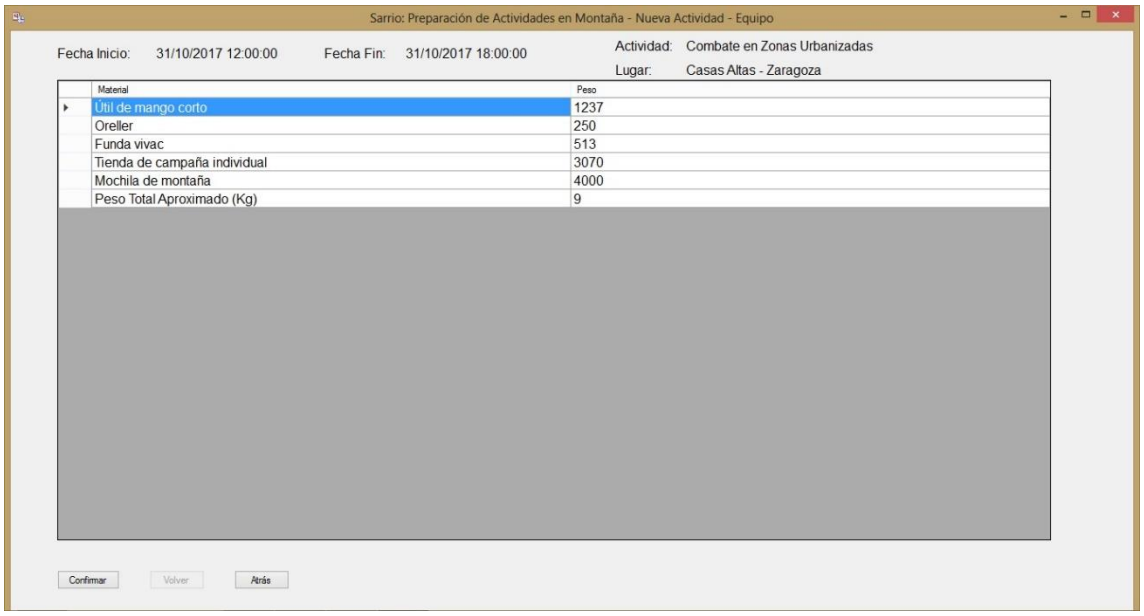


Figura 10 - Anexo A. Ventana de confirmación de equipo y material seleccionado.

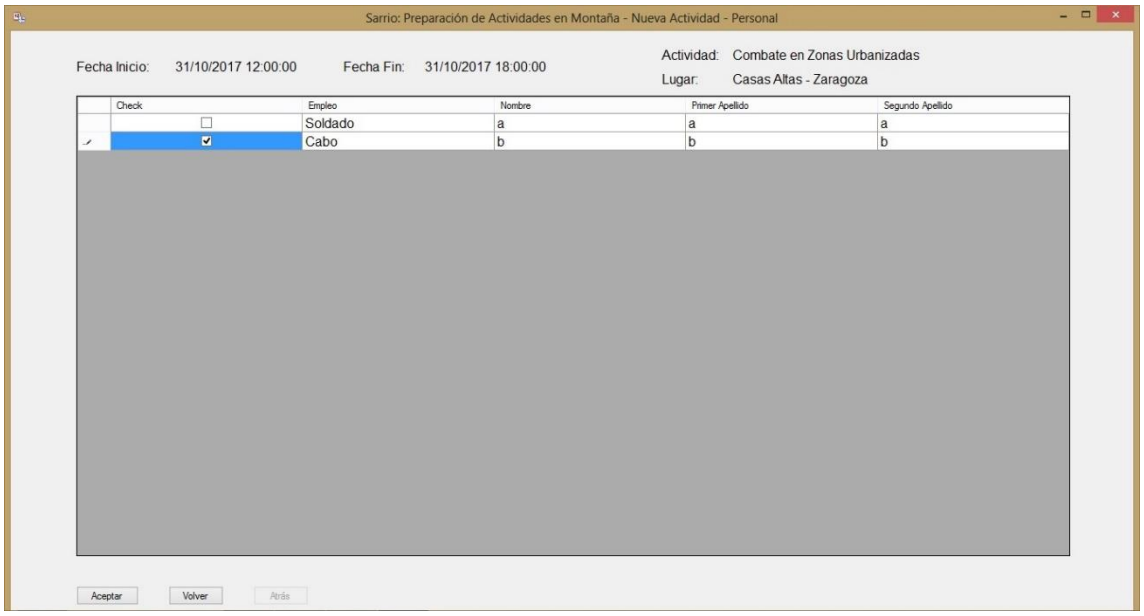


Figura 11 - Anexo A. Ventana de selección de personal.

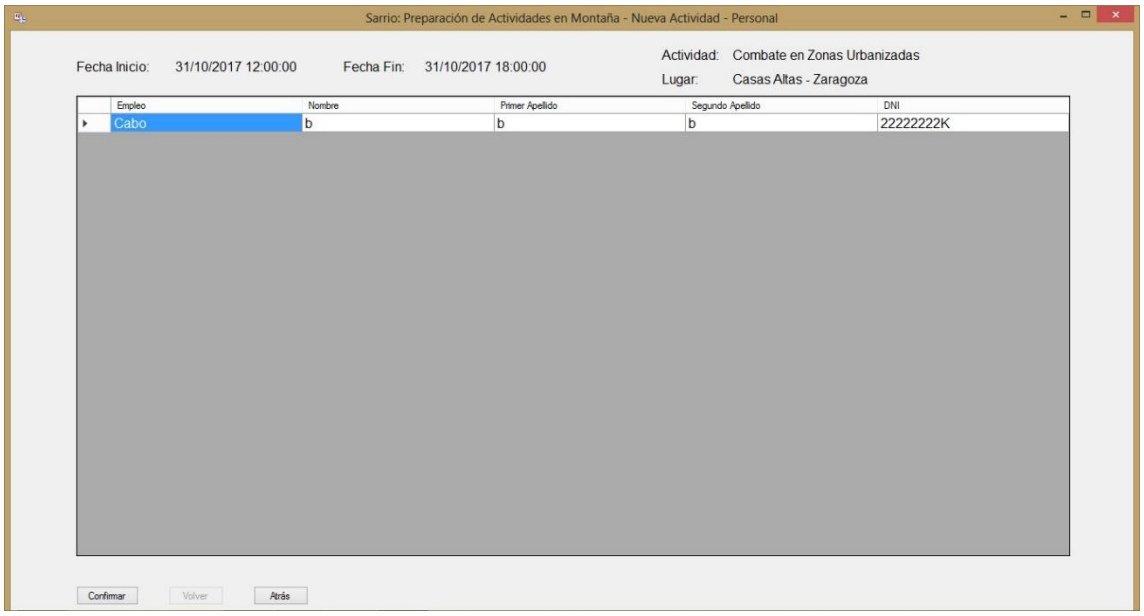


Figura 12 - Anexo A. Ventana de confirmación de personal seleccionado.

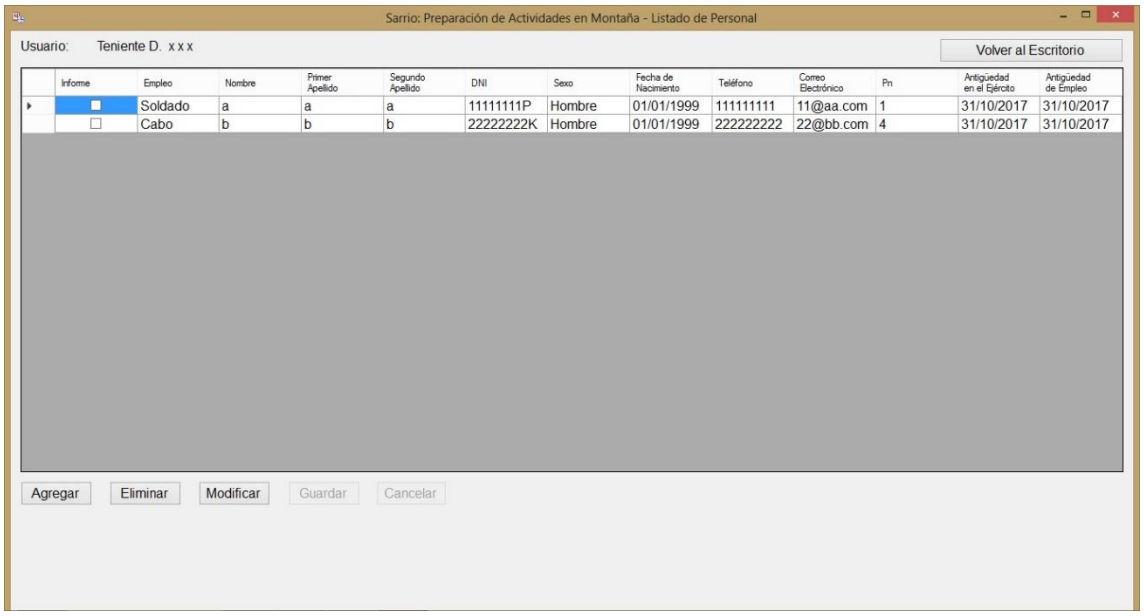


Figura 13 - Anexo A. Ventana "Listado de Personal".

Figura 14 - Anexo A. Ventana "Agregar Personal".

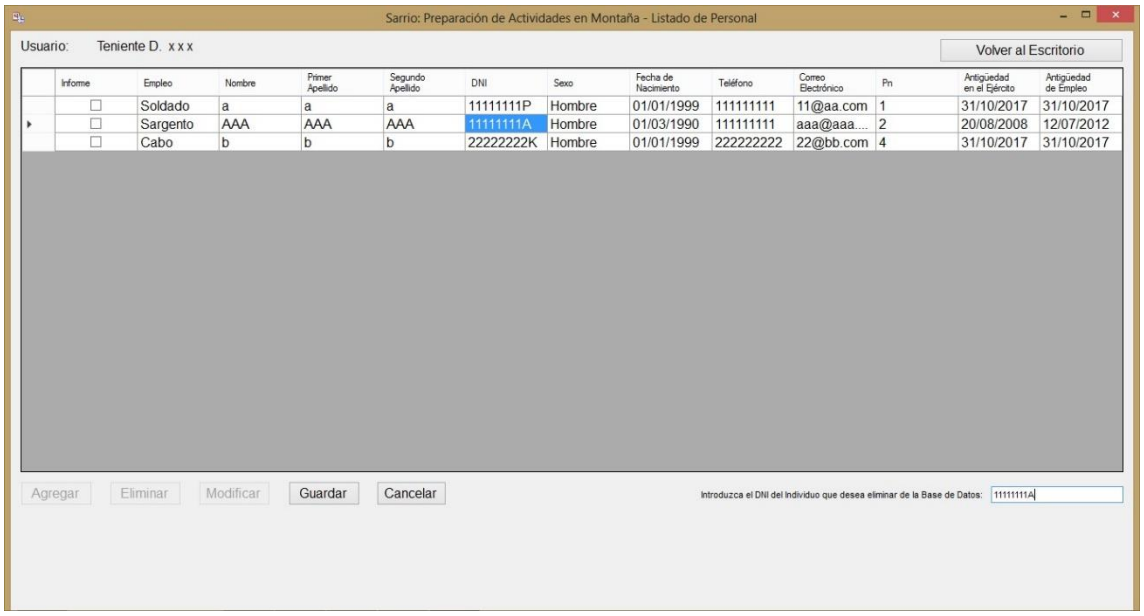


Figura 15 - Anexo A. Opción "Eliminar" un registro de personal.

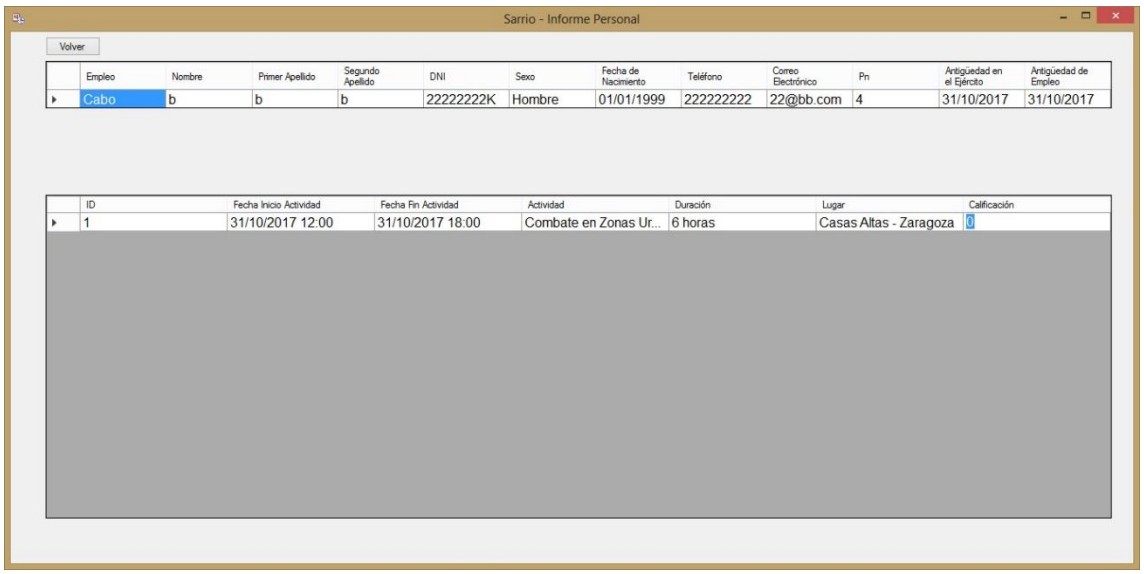


Figura 16 - Anexo A. Ventana "Informe Personal"

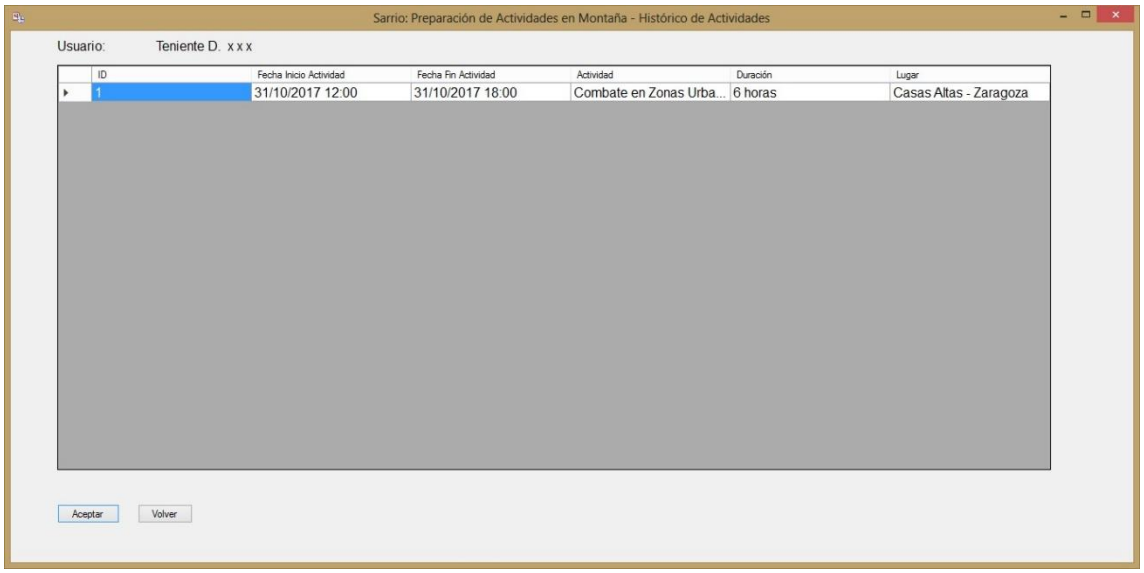


Figura 17 - Anexo A. Ventana "Histórico de Actividades".

Anexo B

Código VB de “Sarrio – Preparación de Actividades en Montaña”

1. Formulario Base

```
Public Class frmBase
    'Para enlazar en cualquier momento un formulario con el Usuario Conectado
    Public usuario As String = ""
    Public ususql As String = ""
    Public dtpIni As Date
    Public dtpFin As Date
    Public actividad As String = ""
    Public duracion As String = ""
    Public lugar As String = ""
    Public Sub frmBase_Activated(sender As Object, e As System.EventArgs)
Handles Me.Activated
        Me.Hide()
        frmEntrada.Show()
        frmEntrada.Focus()
    End Sub
    '  Public Sub cerrar()
    '     If frmEntrada.estado = False Then
    ' Dim q As Integer
    '        q = MsgBox("¿Está seguro de que quiere salir de la aplicación?",
vbYesNo + vbQuestion, "Mensaje")
    '      If q = vbYes Then
    '          Me.Close()
    '        Else
    '          frmEntrada.Show()
    '        End If
    '      End If
    '  End Sub
End Class
```

2. Formulario acceso a la aplicación

```
Imports System.Data.OleDb

Public Class frmEntrada
    'Comandos para el botón "Aceptar"
    Private Sub cmdAceptar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdAceptar.Click
        'Si algún campo no se rellena se abre una ventana emergente de error
        If txtUsuario.Text = "" Or txtContraseña.Text = "" Then
            MsgBox("Se debe introducir datos en todos los campos.",
vbExclamation, "¡Error!")
            txtUsuario.Text = ""
            txtContraseña.Text = ""
            txtUsuario.Focus()
            'Si se introducen datos en los dos campos el programa hace una
conexión a la base de datos para comprobar si el usuario existe
        Else
```

```
        Dim con As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_comunes.accdb")
        Dim pas As String = "SELECT Contraseña FROM usuarios WHERE
Usuario = '" & txtUsuario.Text & "'"
        Dim int As Integer
        Dim intFicha As Integer
        Dim contraseña As String = ""
        int = 0
        Try
            Dim DAP As New OleDbDataAdapter(pas, con)
            Dim DTP As New DataTable
            DAP.Fill(DTP)
            For Each drp As DataRow In DTP.Rows
                contraseña &= drp.Item("Contraseña").ToString
                int = int + 1
            Next
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        'Si el usuario no ha sido encontrado en la base de datos (int=0), quiere
decir que no existe, es decir, nadie se ha registrado con esos datos
        If int = 0 Then
            MsgBox("No se ha encontrado el usuario.", vbExclamation, "¡El
usuario no existe!")
            txtUsuario.Text = ""
            txtContraseña.Text = ""
            txtUsuario.Focus()
            'Si la contraseña de la base de datos difiere de la contraseña
introducida por el usuario, se abre una ventana emergente notificando un error
        ElseIf contraseña <> txtContraseña.Text Then
            MsgBox("El usuario y la contraseña no coinciden.", vbExclamation,
"¡Error!")
            txtContraseña.Text = ""
            txtContraseña.Focus()
            'En esa misma base de datos hay una tabla diferente a la que
almacena el registro del usuario y la contraseña.
            'En dicha tabla se encuentran una serie de datos personales que el
usuario debe rellenar obligatoriamente para acceder al programa.
        Else
            Dim usu As String = "SELECT Usuario FROM datos WHERE Usuario
= '" & txtUsuario.Text & "'"
            Dim usuario As String = ""
            intFicha = 0
            Try
                Dim DAU As New OleDbDataAdapter(usu, con)
                Dim DTU As New DataTable
                DAU.Fill(DTU)
                For Each dru As DataRow In DTU.Rows
                    usuario &= dru.Item("Usuario").ToString
```

```
                intFicha = intFicha + 1
            Next
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        'En caso de que el usuario se hubiera registrado y no hubiera
rellenado sus datos personales, el programa lo detectaría y abriría una ventana
emergente en la que le solicitaría rellenar los datos.
        'Tras pulsar "Aceptar" en dicha ventana emergente se cerraría la
ventana de principal y se abriría la correspondiente para rellenar los datos
personales.
        If intFicha = 0 Then
            MsgBox("El usuario no introdujo sus datos personales tras el
registro." & (Chr(13)) & "Proceda a rellenar la ficha.", vbExclamation, "¡Faltan
datos personales!")
            txtContraseña.Text = ""
            txtUsuario.Focus()
            Me.Hide()
            frmDatosNuevoUsuario.Show()
            'En caso de que sí que se encontraran los datos personales en la
base de datos, el programa almacenaría la información del usuario en memoria
"volátil" para que se viera reflejado qué usuario está conectado.
        Else
            Dim nom As String = "SELECT Empleo, Sexo, Nombre, Apellido1,
Apellido2 FROM datos WHERE Usuario = '" & txtUsuario.Text & "'"
            Dim sexo As String = ""
            Dim empleo As String = ""
            Dim nombre As String = ""
            Dim apellido1 As String = ""
            Dim apellido2 As String = ""
            Try
                Dim DAF As New OleDbDataAdapter(nom, con)
                Dim DTF As New DataTable
                DAF.Fill(DTF)
                For Each drf As DataRow In DTF.Rows
                    sexo &= drf.Item("Sexo").ToString
                    empleo &= drf.Item("Empleo").ToString
                    nombre &= drf.Item("Nombre").ToString
                    apellido1 &= drf.Item("Apellido1").ToString
                    apellido2 &= drf.Item("Apellido2").ToString
                Next
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
            If sexo = "Hombre" Then
                frmBase.usuario = empleo & " D. " & " " & nombre & " " &
apellido1 & " " & apellido2
                frmBase.ususql = usuario
            Else
```

```
                    frmBase.usuario = empleo & " Dña. " & " " & nombre & " " &
apellido1 & " " & apellido2
                    frmBase.ususql = usuario
                End If
                'Comprobación de que se han creado o no se han eliminado las
tablas correspondientes al usuario en la Base de Datos.
                Try
                    Dim DAT As New OleDbDataAdapter(nom, con)
                    Dim DTT As New DataTable
                    DAT.Fill(DTT)
                    For Each drt As DataRow In DTT.Rows

                    Next
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                Me.Close()
                frmEscritorio.Show()
            End If
        End If
        If intFicha <> 0 And int = 0 Then
            'Existen datos en la ficha de un usuario que no existe en la tabla de
cuentas de usuario. Se procede a borrar los datos de la ficha de ese usuario
que no existe.
            con.Open()
            Dim cmd As New OleDb.OleDbCommand("DELETE FROM datos
WHERE Usuario='" & txtUsuario.Text & "'", con)
            cmd.CommandType = CommandType.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Usuario",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Usuario").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@DNI",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@DNI").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Sexo",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Sexo").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Empleo",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Empleo").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Nombre",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Nombre").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido1",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Apellido1").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido2",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Apellido2").Value = ""
```

```
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Unidad",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Unidad").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Compañía",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Compañía").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@FechaNac",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@FechaNac").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEje",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEje").Value = ""
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEmp",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEmp").Value = ""
            cmd.ExecuteNonQuery()
            con.Close()
        End If

    End If
  End Sub

  '"Limpiar" permite que se eliminen todos los caracteres de todos los campos.
  Private Sub cmdLimpiar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdLimpiar.Click
    txtUsuario.Text = ""
    txtContraseña.Text = ""
    txtUsuario.Focus()
  End Sub

  '"Nuevo Usuario" permite el registro de nuevos usuarios.
  Private Sub cmdNuevo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdNuevo.Click
    Me.Close()
    frmNuevo.Show()
  End Sub

  '"Cerrar" permite salir del programa.
  Private Sub cmdCerrar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdCerrar.Click
    Dim dlgCerrar As Integer
    dlgCerrar = MsgBox("¿Está seguro de que desea cerrar el programa?", _
    vbYesNo + vbQuestion, "Mensaje")
    If dlgCerrar = vbYes Then
      Me.Close()
      frmBase.Close()
    End If
  End Sub
```

```
    '   Public estado As Boolean
    ''   Public Sub frmEntrada_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
    '       estado = True
    '  End Sub

    '  Public Sub frmEntrada_Deactivate(sender As System.Object, e As
System.EventArgs) Handles MyBase.Deactivate
    '      estado = False
    '      frmBase.cerrar()
    '   End Sub

End Class
```

3. Formulario para registrar un nuevo usuario

```
Imports System.Data.OleDb

Public Class frmNuevo
    Public frmNuevo As Boolean = False
    Private Sub cmdAceptar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdAceptar.Click
        If txtUsuario.Text = "" Or txtContraseña.Text = "" Or
txtRepContraseña.Text = "" Then
            MsgBox("Se debe introducir datos en todos los campos.",
vbExclamation, "¡Error!")
            txtUsuario.Text = ""
            txtContraseña.Text = ""
            txtRepContraseña.Text = ""
            txtUsuario.Focus()

        Else
            Dim usu As String = "Select Usuario from usuarios where Usuario = '" &
txtUsuario.Text & "'"
            Dim con As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_comunes.accdb")
            Dim int As Integer
            Dim usuario As String = ""
            int = 0
            Try
                Dim DA As New OleDbDataAdapter(usu, con)
                Dim DT As New DataTable
                DA.Fill(DT)
                For Each dr As DataRow In DT.Rows
                    usuario &= dr.Item(0).ToString
                    int = int + 1
                Next
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
```

```
        If txtContraseña.Text <> txtRepContraseña.Text Then
            If usuario <> txtUsuario.Text Then
                MsgBox("Las contraseñas no coinciden.", vbExclamation, "¡Error!")
                txtContraseña.Text = ""
                txtRepContraseña.Text = ""
                txtContraseña.Focus()
            Else
                MsgBox("El usuario ya existe." & (Chr(13)) & "Las contraseñas no
coinciden.", vbExclamation, "¡Error!")
                txtUsuario.Text = ""
                txtContraseña.Text = ""
                txtRepContraseña.Text = ""
                txtContraseña.Focus()
            End If
        ElseIf usuario = txtUsuario.Text Then
            MsgBox("El usuario ya existe.", vbExclamation, "¡Error!")
            txtUsuario.Text = ""
            txtContraseña.Text = ""
            txtRepContraseña.Text = ""
            txtContraseña.Focus()
        ElseIf int = 0 Then
            con.Open()
            Dim cmd As New OleDb.OleDbCommand("INSERT INTO usuarios
VALUES(@Usuario, @Contraseña)", con)
            cmd.CommandType = CommandType.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Usuario",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Usuario").Value = txtUsuario.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Contraseña",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Contraseña").Value = txtContraseña.Text
            cmd.ExecuteNonQuery()
            con.Close()
            MsgBox("Se ha registrado con éxito.", vbOKOnly, Title:="Nuevo
usuario: " + txtUsuario.Text)
            Me.Hide()
            frmNuevo = True
            frmDatosNuevoUsuario.Show()
        End If

    End If

End Sub

Private Sub cmdLimpiar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdLimpiar.Click
    txtUsuario.Text = ""
    txtContraseña.Text = ""
    txtRepContraseña.Text = ""
```

```
        txtUsuario.Focus()
    End Sub

    Public Sub cmdCancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdCancelar.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de que quiere cancelar el registro?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmEntrada.Show()
        End If
    End Sub

End Class
```

4. Formulario para registrar los datos del nuevo usuario

```
Imports System.Data.OleDb
Imports System.Data.SqlClient
Imports System.Data.Sql
Public Class frmDatosNuevoUsuario
    Dim conCom As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_comunes.accdb")
    Dim conInd As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")
    Private Sub frmDatosNuevoUsuario_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        'Variable usuario calculada por defecto, ya sea desde la ventana de
registro de Nuevo Usuario o desde la ventana principal por no haber rellenado
los datos de la ficha cuando se registró.
        If frmNuevo.frmNuevo = True Then
            txtUsuario.Text = frmNuevo.txtUsuario.Text
        Else
            txtUsuario.Text = frmEntrada.txtUsuario.Text
        End If
        dtpNac.Value = "#" & Today.Day & "/" & Today.Month & "/" & Today.Year -
18 & "#"
        dtpAntEje.Value = "#" & Today.Day & "/" & Today.Month & "/" &
Today.Year & "#"
        dtpAntEmp.Value = "#" & Today.Day & "/" & Today.Month & "/" &
Today.Year & "#"
        txtDNI.Select()
    End Sub
```

```
    'DNI
    '-------------------------------------------------------------------------------------------
    Private Sub txtDNI_TextChanged(sender As System.Object, e As
System.EventArgs) Handles txtDNI.TextChanged
        'Cuando el texto cambia en el recuadro en el que se introduce el DNI se
comprueban los datos que se están introduciendo con los que hay en la Base
de Datos.
        Dim DNI As String = "SELECT DNI FROM datos"

        Dim DNIusuario As String = ""
        Try
            Dim DA As New OleDbDataAdapter(DNI, conCom)
            Dim DT As New DataTable
            DA.Fill(DT)
            For Each dr As DataRow In DT.Rows
                DNIusuario &= dr.Item("DNI").ToString
                'En el caso de que coincida el DNI introducido con uno de la Base de
Datos, se activa un mensaje de error y se borra el DNI introducido.
                If txtDNI.Text = DNIusuario Then
                    MsgBox("El DNI introducido ya está registrado.", vbExclamation,
"¡Error!")
                    txtDNI.Text = ""
                    txtDNI.Focus()
                End If
            Next
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub txtDNI_Leave(sender As System.Object, e As
System.EventArgs) Handles txtDNI.Leave
        Dim i As Integer = 0
        'Una vez se deselecciona el recuadro para introducir el DNI, el programa
comprueba si dicho campo contiene 9 caracteres (8 primeros como números y
el último como letra) o si el campo está vacío en caso de que se desee rellenar
después de rellenar otros.
        'En caso contrario, se activa un mensaje de error.
        If txtDNI.Text.Length <> 9 And txtDNI.Text <> "" Then
            MsgBox("Introduzca un DNI válido.", vbExclamation, "¡Error!")
            txtDNI.Text = ""
            txtDNI.Focus()
            i = 0
        End If
        If txtDNI.TextLength = 9 Then
            If txtDNI.Text(8) = "1" Or txtDNI.Text(8) = "2" Or txtDNI.Text(8) = "3" Or
txtDNI.Text(8) = "4" Or txtDNI.Text(8) = "5" Or txtDNI.Text(8) = "6" Or
txtDNI.Text(8) = "7" Or txtDNI.Text(8) = "8" Or txtDNI.Text(8) = "9" Or
txtDNI.Text(8) = "0" Then
```

```
            MsgBox("El último carácter del DNI debe ser una letra.",
vbExclamation, "¡Error!")
            txtDNI.Text = ""
            txtDNI.Focus()
            i = 0
        End If
    End If
    'El programa detecta si se escribe un espacio en blanco y activa un
mensaje de error.
    If txtDNI.TextLength <> 0 Then
        For i = 0 To (txtDNI.MaxLength() - 1)
            If txtDNI.Text(i) = " " Then
                MsgBox("No se puede introducir espacios en blanco.",
vbExclamation, "¡Error!")
                txtDNI.Text = ""
                txtDNI.Focus()
                Return
            ElseIf txtDNI.Text(i) = "º" Or txtDNI.Text(i) = "ª" Or txtDNI.Text(i) = "\"
Or txtDNI.Text(i) = "!" Or txtDNI.Text(i) = "|" Or txtDNI.Text(i) = "@" Or
txtDNI.Text(i) = "·" Or txtDNI.Text(i) = "#" Or txtDNI.Text(i) = "$" Or
txtDNI.Text(i) = "%" Or txtDNI.Text(i) = "&" Or txtDNI.Text(i) = "¬" Or
txtDNI.Text(i) = "/" Or txtDNI.Text(i) = "(" Or txtDNI.Text(i) = ")" Or txtDNI.Text(i)
= "=" Or txtDNI.Text(i) = "'" Or txtDNI.Text(i) = "?" Or txtDNI.Text(i) = "¡" Or
txtDNI.Text(i) = "¿" Or txtDNI.Text(i) = "€" Or txtDNI.Text(i) = "`" Or
txtDNI.Text(i) = "^" Or txtDNI.Text(i) = "[" Or txtDNI.Text(i) = "+" Or
txtDNI.Text(i) = "*" Or txtDNI.Text(i) = "]" Or txtDNI.Text(i) = "´" Or txtDNI.Text(i)
= "¨" Or txtDNI.Text(i) = "{" Or txtDNI.Text(i) = "Ç" Or txtDNI.Text(i) = "ç" Or
txtDNI.Text(i) = "}" Or txtDNI.Text(i) = "<" Or txtDNI.Text(i) = ">" Or
txtDNI.Text(i) = "," Or txtDNI.Text(i) = ";" Or txtDNI.Text(i) = "." Or txtDNI.Text(i)
= ":" Or txtDNI.Text(i) = "-" Or txtDNI.Text(i) = "_" Or txtDNI.Text(i) = "'" Then
                MsgBox("Solo puede introducirse números y en la última posición
una letra.", vbExclamation, "¡Error!")
                txtDNI.Text = ""
                txtDNI.Focus()
                Return
            End If
        Next
    End If
End Sub
'-----------------------------------------------------------------------------------------

'"Data Time Picker,s"
'-----------------------------------------------------------------------------------------
Private Sub dtpAntEje_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpAntEje.ValueChanged
    If dtpAntEje.Value < dtpNac.Value Then
        MsgBox("La Antigüedad en el Ejército no puede ser anterior a la fecha
de nacimiento.", vbExclamation, "¡Error!")
        dtpAntEje.Value = Today
    End If
```

```
    End Sub

    Private Sub dtpAntEmp_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpAntEmp.ValueChanged
        If dtpAntEmp.Value < dtpNac.Value Then
            MsgBox("La Antigüedad de Empleo no puede ser anterior a la fecha de
nacimiento.", vbExclamation, "¡Error!")
            dtpAntEmp.Value = dtpAntEje.Value
        End If
        If dtpAntEmp.Value < dtpAntEje.Value Then
            MsgBox("La Antigüedad de Empleo no puede ser anterior a la
Antigüedad en el Ejército.", vbExclamation, "¡Error!")
            dtpAntEmp.Value = dtpAntEje.Value
        End If
    End Sub
    '-------------------------------------------------------------------------------------------
    'Botones ("Aceptar" y "Cancelar")
    '-------------------------------------------------------------------------------------------
    Private Sub cmdAceptar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAceptar.Click

        If txtDNI.Text = "" Or cmbSexo.Text = "" Or cmbEmpleo.Text = "" Or
txtNombre.Text = "" Or txtApellido1.Text = "" Or txtApellido2.Text = "" Or
txtUd.Text = "" Or txtCia.Text = "" Then
            MsgBox("Se debe introducir datos en todos los campos.",
vbExclamation, "¡Error!")
        Else
            Dim usu As String = "SELECT Usuario FROM datos WHERE Usuario =
'" & txtUsuario.Text & "'"
            Dim int As Integer
            Dim usuario As String = ""

            int = 0

            Try
                Dim DA As New OleDbDataAdapter(usu, conCom)
                Dim DT As New DataTable

                DA.Fill(DT)

                For Each dr As DataRow In DT.Rows
                    usuario &= dr.Item(0).ToString
                    int = int + 1
                Next
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
```

```
        If int = 0 Then
            conCom.Open()
            Dim cmd As New OleDb.OleDbCommand("INSERT INTO datos
VALUES(@Usuario, @DNI, @Sexo, @Empleo, @Nombre, @Apellido1,
@Apellido2, @Unidad, @Compañía, @FechaNac, @AntEje, @AntEmp)",
conCom)
            cmd.CommandType = CommandType.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Usuario",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Usuario").Value = txtUsuario.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@DNI",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@DNI").Value = txtDNI.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Sexo",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Sexo").Value = cmbSexo.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Empleo",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Empleo").Value = cmbEmpleo.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Nombre",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Nombre").Value = txtNombre.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido1",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Apellido1").Value = txtApellido1.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido2",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Apellido2").Value = txtApellido2.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Unidad",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Unidad").Value = txtUd.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Compañía",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Compañía").Value = txtCia.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@FechaNac",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@FechaNac").Value = dtpNac.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEje",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEje").Value = dtpAntEje.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEmp",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEmp").Value = dtpAntEmp.Text
            cmd.ExecuteNonQuery()
            conCom.Close()

            'Se crea una tabla con el nombre de usuario y los campos del listado
del personal
            conInd.Open()
```

```
            Dim cmdList As New OleDb.OleDbCommand("CREATE TABLE " &
txtUsuario.Text & "_listado(" & (Chr(13)) & "ID INT," & (Chr(13)) & "[Check]
BIT," & (Chr(13)) & "Empleo VARCHAR (20)," & (Chr(13)) & "Nombre
VARCHAR (20)," & (Chr(13)) & "[Primer Apellido] VARCHAR (20)," & (Chr(13))
& "[Segundo Apellido] VARCHAR (20)," & (Chr(13)) & "DNI VARCHAR (9)," &
(Chr(13)) & "Sexo VARCHAR (6)," & (Chr(13)) & "[Fecha de Nacimiento]
DATE," & (Chr(13)) & "[Teléfono] INT," & (Chr(13)) & "Pn INT," & (Chr(13)) &
"[Correo Electrónico] VARCHAR (20)," & (Chr(13)) & "[Antigüedad en el
Ejército] DATE, " & (Chr(13)) & "[Antigüedad de Empleo] DATE," & (Chr(13)) &
"PRIMARY KEY (DNI))", conInd)
            cmdList.CommandType = CommandType.Text
            cmdList.ExecuteNonQuery()
            Dim cmdHist As New OleDb.OleDbCommand("CREATE TABLE " &
txtUsuario.Text & "_historico(" & (Chr(13)) & "ID INT," & (Chr(13)) & "[Fecha
Inicio Actividad] DATETIME," & (Chr(13)) & "[Fecha Fin Actividad] DATETIME,"
& (Chr(13)) & "Actividad VARCHAR (50)," & (Chr(13)) & "Duración VARCHAR
(50)," & (Chr(13)) & "Lugar VARCHAR (50)," & (Chr(13)) & "PRIMARY KEY
(ID))", conInd)
            cmdHist.CommandType = CommandType.Text
            cmdHist.ExecuteNonQuery()
            conInd.Close()

            If cmbSexo.Text = "Hombre" Then
                MsgBox("Sus datos han sido registrados con éxito.", vbOKOnly,
Title:="Nuevo usuario: " & cmbEmpleo.Text & " D. " & txtNombre.Text & " " &
txtApellido1.Text & " " & txtApellido2.Text)
                Me.Close()
                frmNuevo.Close()
                frmEntrada.Show()
            Else
                MsgBox("Sus datos han sido registrados con éxito.", vbOKOnly,
Title:="Nuevo usuario: " & cmbEmpleo.Text & " Dña. " & txtNombre.Text & " " &
txtApellido1.Text & " " & txtApellido2.Text)
                Me.Close()
                frmNuevo.Close()
                frmEntrada.Show()
            End If
        End If

    End If

    End Sub

    Public Sub cmdCancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdCancelar.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de que quiere cancelar el registro de
sus datos personales?" & (Chr(13)) & "No podrá acceder a la aplicación hasta
que haya rellenado la ficha.", vbYesNo + vbQuestion, "¡Advertencia!")
        If dlgCerrar = vbYes Then
```

```
            Me.Close()
            frmEntrada.Show()
        End If
    End Sub
End Class
```

5. Formulario del escritorio

```
Public Class frmEscritorio

    Private Sub frmEscritorio_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        txtUsuario.Text = frmBase.usuario
    End Sub

    Private Sub cmdNuevaActividad_Click(sender As System.Object, e As
System.EventArgs) Handles cmdNuevaActividad.Click
        Me.Close()
        frmActividad.Show()
    End Sub

    Private Sub cmdListado_Click(sender As System.Object, e As
System.EventArgs) Handles cmdListado.Click
        Me.Close()
        frmListado.Show()
    End Sub

    Private Sub cmdHistorico_Click(sender As System.Object, e As
System.EventArgs) Handles cmdHistorico.Click
        Me.Close()
        frmHistorico.Show()
    End Sub

    Private Sub cmdCerrarSesion_Click(sender As System.Object, e As
System.EventArgs) Handles cmdCerrarSesion.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de que desea cerrar sesión?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmEntrada.Show()
            frmBase.usuario = ""
        End If
    End Sub
End Class
```

6. Formulario para registrar nueva actividad

```
Public Class frmActividad
    Dim duracion As Long
    Dim horaFin As DateTime
    Dim horaIni As DateTime

    Public Sub frmActividad_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load

        dtpHoraFin.Value = CDate(FormatDateTime("#" & Today & "#",
DateFormat.GeneralDate) & " " & FormatDateTime("#12:00:00#",
DateFormat.LongTime))
        dtpHoraIni.Value = CDate(FormatDateTime("#" & Today & "#",
DateFormat.GeneralDate) & " " & FormatDateTime("#12:00:00#",
DateFormat.LongTime))
        dtpFin.Value = Today
        dtpIni.Value = Today
        refresh_date()

    End Sub

    Public Sub refresh_date()
        horaFin = dtpHoraFin.Value
        horaIni = dtpHoraIni.Value
        duracion = DateDiff(DateInterval.DayOfYear, CDate(dtpIni.Value),
CDate(dtpFin.Value), FirstDayOfWeek.Monday, FirstWeekOfYear.Jan1)


        If CInt(duracion) < 0 Then
            MsgBox("La fecha de finalización de la actividad no puede ser anterior a
la fecha de inicio.", vbExclamation, "¡Error!")
            dtpFin.Value = dtpIni.Value


        ElseIf CInt(duracion) >= 0 And CInt(duracion) < 1 Then
            If CInt(DateDiff(DateInterval.Hour, horaIni, horaFin)) < 0 Then
                MsgBox("La hora de finalización de la actividad no puede ser anterior
a la hora de inicio para la misma fecha.", vbExclamation, "¡Error!")
                dtpHoraFin.Value = dtpHoraIni.Value
            ElseIf CInt(DateDiff(DateInterval.Hour, horaIni, horaFin)) >= 0 And
CInt(DateDiff(DateInterval.Hour, horaIni, horaFin)) < 1 Then
                txtDuracion.Text = "Menos de 1 hora"
                frmBase.dtpIni = FormatDateTime(dtpHoraIni.Value,
DateFormat.GeneralDate)
                frmBase.dtpFin = FormatDateTime(dtpHoraFin.Value,
DateFormat.GeneralDate)
                frmBase.duracion = txtDuracion.Text
            ElseIf CInt(DateDiff(DateInterval.Hour, horaIni, horaFin)) = 1 Then
```

```
            txtDuracion.Text = DateDiff(DateInterval.Hour, horaIni,
horaFin).ToString & " hora"
            frmBase.dtpIni = FormatDateTime(dtpHoraIni.Value,
DateFormat.GeneralDate)
            frmBase.dtpFin = FormatDateTime(dtpHoraFin.Value,
DateFormat.GeneralDate)
            frmBase.duracion = txtDuracion.Text
        ElseIf CInt(DateDiff(DateInterval.Hour, horaIni, horaFin)) > 1 Then
            txtDuracion.Text = DateDiff(DateInterval.Hour, horaIni,
horaFin).ToString & " horas"
            frmBase.dtpIni = FormatDateTime(dtpHoraIni.Value,
DateFormat.GeneralDate)
            frmBase.dtpFin = FormatDateTime(dtpHoraFin.Value,
DateFormat.GeneralDate)
            frmBase.duracion = txtDuracion.Text
        End If
    ElseIf CInt(duracion) = 1 Then
        txtDuracion.Text = duracion & " día"
        frmBase.dtpIni = dtpIni.Value
        frmBase.dtpFin = dtpFin.Value
        frmBase.duracion = txtDuracion.Text
    ElseIf CInt(duracion) > 1 Then
        txtDuracion.Text = duracion & " días"
        frmBase.dtpIni = dtpIni.Value
        frmBase.dtpFin = dtpFin.Value
        frmBase.duracion = txtDuracion.Text
    End If
End Sub

Private Sub dtpIni_Leave(sender As System.Object, e As System.EventArgs)
Handles dtpIni.Leave
    If CDate(dtpFin.Value) < CDate(dtpIni.Value) Then
        dtpFin.Value = dtpIni.Value
        refresh_date()
    End If
End Sub

Private Sub dtpIni_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpIni.ValueChanged
    txtDuracion.Select()
    refresh_date()
End Sub

Private Sub dtpHoraIni_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpHoraIni.ValueChanged
    txtDuracion.Select()
    refresh_date()
End Sub
```

```
    Private Sub dtpHoraIni_Leave(sender As System.Object, e As
System.EventArgs) Handles dtpHoraIni.Leave
        If CInt(duracion) < 1 And CDate(dtpHoraFin.Value) <
CDate(dtpHoraIni.Value) Then
            dtpHoraFin.Value = dtpHoraIni.Value
            txtDuracion.Select()
            refresh_date()
        End If
    End Sub

    Private Sub dtpFin_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpFin.ValueChanged
        txtDuracion.Select()
        refresh_date()
    End Sub

    Private Sub dtpHoraFin_Leave(sender As System.Object, e As
System.EventArgs) Handles dtpHoraFin.Leave
        txtDuracion.Select()
        refresh_date()
    End Sub

    Private Sub dtpHoraFin_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpHoraFin.ValueChanged
        txtDuracion.Select()
        refresh_date()
    End Sub
    Private Sub cmdAceptar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAceptar.Click
        frmBase.actividad = cmbActividad.Text
        frmBase.lugar = txtLugar.Text
        If cmbActividad.Text = "Combate en Zonas Urbanizadas" Or
cmbActividad.Text = "Escalada" Or cmbActividad.Text = "Esquí" Or
cmbActividad.Text = "Explosivos" Or cmbActividad.Text = "Marcha" _
Or cmbActividad.Text = "Práctica de montaje pasamanos" Or
cmbActividad.Text = "Práctica de rescate con LEVA / ARVA" _
Or cmbActividad.Text = "Práctica de rescate con sondas" Or cmbActividad.Text
= "Práctica de rescate con UT" Or cmbActividad.Text = "Rapell" _
Or cmbActividad.Text = "Recorrido Topográfico" Or cmbActividad.Text =
"Semipermanente" Or cmbActividad.Text = "Tema Táctico " Or
cmbActividad.Text = "Tiro" _
Or cmbActividad.Text = "Otro tipo de Actividad" Then
            Me.Close()
            frmEquipo.Show()
        Else
            MsgBox("Elija una actividad de la lista desplegable.", vbExclamation,
"¡Error!")
        End If
    End Sub
```

```
    Private Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de desea volver al Escritorio?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmEscritorio.Show()
        End If
    End Sub
End Class
```

7. Formulario para seleccionar equipo de actividad a realizar

```
Imports System.Data.OleDb
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Public Class frmEquipo
    Dim con As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas.accdb")

    Public Sub limpiar()
        Dim DAL As New OleDbDataAdapter("UPDATE [Material] SET [Check] =
False", con)
        Dim DSL As New DataSet
        DAL.Fill(DSL, "Material")
        dgvListado.DataSource = DSL.Tables("Material")
    End Sub

    Public Sub tabla()
        Dim DAT As New OleDbDataAdapter("SELECT [Material], [Peso], [Check],
[Tipo], [Fase] FROM Material ORDER BY ID ASC", con)
        Dim DST As New DataSet
        DAT.Fill(DST, "Material")
        dgvListado.DataSource = DST.Tables("Material")
    End Sub

    Public Sub equipo_seleccionado()
        Dim DAC As New OleDbDataAdapter("SELECT [Material], [Peso] FROM
Material WHERE [Check] = True ORDER BY ID ASC", con)
        Dim DSC As New DataSet
        DAC.Fill(DSC, "Material")
        dgvListado.DataSource = DSC.Tables("Material")
        Dim suma As Decimal
        For Each dr As DataRow In DSC.Tables("Material").Rows
            suma = suma + CDec(dr.Item("Peso").ToString)
        Next
        Dim cmd As New OleDb.OleDbCommand("INSERT INTO [Material] ([ID],
[Material], [Peso], [Check], [Tipo], [Fase]) VALUES('9999', 'Peso Total
```

```
Aproximado (Kg)',  '" & CDec(suma / 1000) & "', True, 'Equipo', 'Indiferente')",
con)
        cmd.CommandType = CommandType.Text
        con.Open()
        cmd.ExecuteNonQuery()
        con.Close()
        Dim DAE As New OleDbDataAdapter("SELECT [Material], [Peso] FROM
Material WHERE [Check] = True ORDER BY ID ASC", con)
        Dim DSE As New DataSet
        DAE.Fill(DSE, "Material")
        dgvListado.DataSource = DSE.Tables("Material")
        con.Open()
        Dim del As New OleDb.OleDbCommand("DELETE * FROM [Material]
WHERE ID = 9999", con)
        del.CommandType = CommandType.Text
        del.ExecuteNonQuery()
        con.Close()
    End Sub

    Public Sub frmEquipo_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        limpiar()
        tabla()
        cmdAtras.Enabled = False
        txtIni.Text = frmBase.dtpIni
        txtFin.Text = frmBase.dtpFin
        txtActividad.Text = frmBase.actividad
        txtLugar.Text = frmBase.lugar
        cmdConfirmar.Hide()

        dgvListado.Columns.Item("Material").ReadOnly = True
        dgvListado.Columns.Item("Peso").ReadOnly = True
        dgvListado.Columns.Item("Tipo").ReadOnly = True
        dgvListado.Columns.Item("Fase").ReadOnly = True
    End Sub

    Public Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de desea volver al Escritorio?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmActividad.Show()
        End If
    End Sub


    Public Sub cmdAceptar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAceptar.Click
```

```
        equipo_seleccionado()
        cmdAtras.Enabled = True
        cmdVolver.Enabled = False
        cmdAceptar.Hide()
        cmdConfirmar.Show()
    End Sub

    Private Sub dgvListado_CellEndEdit(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellEndEdit

        Dim j As Integer = dgvListado.CurrentRow.Index

        Dim upd As New OleDb.OleDbCommand("UPDATE [Material] SET
[Check] = " & dgvListado.Item("Check", j).EditedFormattedValue.ToString & "
WHERE [ID] = " & j + 1 & "", con)
        con.Open()
        upd.CommandType = CommandType.Text
        upd.ExecuteNonQuery()
        con.Close()
    End Sub

    Private Sub cmdAtras_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAtras.Click
        cmdAtras.Enabled = False
        cmdVolver.Enabled = True
        tabla()
        cmdAceptar.Show()
        cmdConfirmar.Hide()
        dgvListado.Columns.Item("Material").ReadOnly = True
        dgvListado.Columns.Item("Peso").ReadOnly = True
        dgvListado.Columns.Item("Tipo").ReadOnly = True
        dgvListado.Columns.Item("Fase").ReadOnly = True
    End Sub

    Private Sub cmdConfirmar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdConfirmar.Click
        Me.Close()
        frmPersonal.Show()
    End Sub
End Class
```

8. Formulario para seleccionar personal de actividad a realizar

```
Imports System.Data.OleDb
Public Class frmPersonal
    Dim conInd As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")

    Public Sub limpiar()
        Dim DAL As New OleDbDataAdapter("UPDATE " & frmBase.ususql &
"_listado " & "SET [Check] = False", conInd)
        Dim DSL As New DataSet
        DAL.Fill(DSL, frmBase.ususql & "_listado")
        dgvListado.DataSource = DSL.Tables(frmBase.ususql & "_listado")
    End Sub

    Public Sub tabla()
        Dim DAT As New OleDbDataAdapter("SELECT [Check], [Empleo],
[Nombre], [Primer Apellido], [Segundo Apellido]  FROM " & frmBase.ususql &
"_listado" & " ORDER BY Pn ASC", conInd)
        Dim DST As New DataSet
        DAT.Fill(DST, frmBase.ususql & "_listado")
        dgvListado.DataSource = DST.Tables(frmBase.ususql & "_listado")
    End Sub

    Public Sub personal_seleccionado()
        Dim DAC As New OleDbDataAdapter("SELECT [Empleo], [Nombre],
[Primer Apellido], [Segundo Apellido], [DNI] FROM " & frmBase.ususql &
"_listado" & " WHERE [Check] = True ORDER BY Pn ASC", conInd)
        Dim DSC As New DataSet
        DAC.Fill(DSC, frmBase.ususql & "_listado")
        dgvListado.DataSource = DSC.Tables(frmBase.ususql & "_listado")
    End Sub

    Public Sub frmPersonal_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        limpiar()
        tabla()
        cmdAtras.Enabled = False
        txtIni.Text = frmBase.dtpIni
        txtFin.Text = frmBase.dtpFin
        txtActividad.Text = frmBase.actividad
        txtLugar.Text = frmBase.lugar
        cmdConfirmar.Hide()
        dgvListado.Columns.Item("Empleo").ReadOnly = True
        dgvListado.Columns.Item("Nombre").ReadOnly = True
        dgvListado.Columns.Item("Primer Apellido").ReadOnly = True
        dgvListado.Columns.Item("Segundo Apellido").ReadOnly = True
    End Sub
```

```
    Public Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de desea volver al Escritorio?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmEquipo.Show()
        End If
    End Sub

    Public Sub cmdAceptar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAceptar.Click
        personal_seleccionado()
        cmdAtras.Enabled = True
        cmdVolver.Enabled = False
        cmdAceptar.Hide()
        cmdConfirmar.Show()
    End Sub

    Private Sub dgvListado_CellEndEdit(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellEndEdit
        Dim j As Integer = dgvListado.CurrentRow.Index
        Dim upd As New OleDb.OleDbCommand("UPDATE " & frmBase.ususql &
"_listado SET [Check] = " & dgvListado.Item("Check",
j).EditedFormattedValue.ToString & " WHERE [ID] = " & j + 1 & "", conInd)
        conInd.Open()
        upd.CommandType = CommandType.Text
        upd.ExecuteNonQuery()
        conInd.Close()
    End Sub
    Private Sub cmdAtras_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAtras.Click
        cmdAtras.Enabled = False
        cmdVolver.Enabled = True
        tabla()
        cmdAceptar.Show()
        cmdConfirmar.Hide()
    End Sub

    Private Sub cmdConfirmar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdConfirmar.Click
        Me.Close()
        frmEscritorio.Show()
        Dim id As Integer = 0
        Dim DA As New OleDbDataAdapter("SELECT COUNT (ID) FROM " &
frmBase.ususql & "_historico", conInd)
        Dim DT As New DataTable
        DA.Fill(DT)
```

```
        For Each dr As DataRow In DT.Rows
            id = CInt(dr.Item(0).ToString)
        Next
        Dim uph As New OleDb.OleDbCommand("INSERT INTO " &
frmBase.ususql & "_historico VALUES(" & id + 1 & ", '" & frmBase.dtpIni & "', '"
& frmBase.dtpFin & "', '" & frmBase.actividad & "', '" & frmBase.duracion & "', '"
& frmBase.lugar & "')", conInd)
        conInd.Open()
        uph.CommandType = CommandType.Text
        uph.ExecuteNonQuery()
        Dim DNI As String = ""
        Dim DAD As New OleDbDataAdapter("SELECT DNI FROM " &
frmBase.ususql & "_listado WHERE Check = True", conInd)
        Dim DTD As New DataTable
        DAD.Fill(DTD)
        For Each drd As DataRow In DTD.Rows
            DNI = drd.Item(0).ToString
            Dim ide As New OleDb.OleDbCommand("INSERT INTO " & DNI &
"_informe VALUES(" & id + 1 & ", '" & frmBase.dtpIni & "', '" & frmBase.dtpFin &
"', '" & frmBase.actividad & "', '" & frmBase.duracion & "', '" & frmBase.lugar & "',
'0')", conInd)
            ide.CommandType = CommandType.Text
            ide.ExecuteNonQuery()
        Next
        conInd.Close()
    End Sub
```

9. Formulario listado de personal

```
Option Explicit On
Imports System.Data.OleDb
Imports System.Data.SqlClient

Public Class frmListado
    Dim conCom As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_comunes.accdb")
    Dim conInd As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")
    Dim intEliminar As Boolean = False
    Dim intModificar As Boolean = False
    Public Sub limpiar()
        Dim DAL As New OleDbDataAdapter("UPDATE " & frmBase.ususql &
"_listado " & "SET [Check] = False", conInd)
        Dim DSL As New DataSet
        DAL.Fill(DSL, frmBase.ususql & "_listado")
        dgvListado.DataSource = DSL.Tables(frmBase.ususql & "_listado")
```

```
    End Sub
    Public Sub actualizar_tabla()

        Dim DA As New OleDbDataAdapter("SELECT [Check] As [Informe],
[Empleo], [Nombre], [Primer Apellido], [Segundo Apellido], [DNI], [Sexo], [Fecha
de Nacimiento], [Teléfono], [Correo Electrónico], [Pn], [Antigüedad en el
Ejército], [Antigüedad de Empleo] FROM " & frmBase.ususql & "_listado
ORDER BY ID ASC", conInd)
        Dim DS As New DataSet
        DA.Fill(DS, frmBase.ususql & "_listado")
        dgvListado.DataSource = DS.Tables(frmBase.ususql & "_listado")

        If intEliminar = False Then
            lblDNI.Hide()
            txtDNI.Hide()
            txtDNI.Text = ""
        End If
        dgvListado.Columns.Item(0).ReadOnly = False
        dgvListado.Columns.Item("Empleo").ReadOnly = True
        dgvListado.Columns.Item("Nombre").ReadOnly = True
        dgvListado.Columns.Item("Primer Apellido").ReadOnly = True
        dgvListado.Columns.Item("Segundo Apellido").ReadOnly = True
        dgvListado.Columns.Item("DNI").ReadOnly = True
        dgvListado.Columns.Item("Sexo").ReadOnly = True
        dgvListado.Columns.Item("Fecha de Nacimiento").ReadOnly = True
        dgvListado.Columns.Item("Teléfono").ReadOnly = True
        dgvListado.Columns.Item("Correo Electrónico").ReadOnly = True
        dgvListado.Columns.Item("Pn").ReadOnly = True
        dgvListado.Columns.Item("Antigüedad en el Ejército").ReadOnly = True
        dgvListado.Columns.Item("Antigüedad de Empleo").ReadOnly = True
    End Sub

    Private Sub frmListado_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        limpiar()
        txtUsuario.Text = frmBase.usuario
        cmdGuardar.Enabled = False
        cmdCancelar.Enabled = False
        lblDNI.Hide()
        txtDNI.Hide()
        actualizar_tabla()
    End Sub

    Private Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        Me.Close()
        frmEscritorio.Show()
    End Sub
```

```
    Public Sub cmdAgregar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAgregar.Click
        frmAgregar.Show()
        Me.Hide()
    End Sub

    Private Sub cmdEliminar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdEliminar.Click
        cmdAgregar.Enabled = False
        cmdEliminar.Enabled = False
        cmdModificar.Enabled = False
        cmdGuardar.Enabled = True
        cmdCancelar.Enabled = True
        intEliminar = True
        lblDNI.Show()
        txtDNI.Show()
        txtDNI.Select()
    End Sub

    Private Sub cmdModificar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdModificar.Click
        cmdAgregar.Enabled = False
        cmdEliminar.Enabled = False
        cmdModificar.Enabled = False
        cmdGuardar.Enabled = True
        cmdCancelar.Enabled = True
        dgvListado.ReadOnly = False
        intModificar = True
    End Sub

    Public Sub dgvListado_RowValidated(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.RowValidated
    End Sub

    Private Sub cmdGuardar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdGuardar.Click
            If intEliminar = True Then
                Dim DNI As String = "SELECT DNI FROM " & frmBase.ususql &
"_listado" & (Chr(13)) & "WHERE DNI = '" & txtDNI.Text & "'"
                Dim int As Integer = 0
                Dim ident As String = ""
                Dim i As Integer = 0
                Try
                    Dim DAD As New OleDbDataAdapter(DNI, conInd)
                    Dim DTD As New DataTable

                    DAD.Fill(DTD)

                    For Each drd As DataRow In DTD.Rows
```

```
                    ident &= drd.Item(0).ToString
                    int = int + 1
                Next
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
            If int = 1 Then
                conInd.Open()
                Dim cmd As New OleDb.OleDbCommand("DELETE * FROM " &
frmBase.ususql & "_listado" & (Chr(13)) & "WHERE DNI = '" & txtDNI.Text & "'",
conInd)
                cmd.CommandType = CommandType.Text
                cmd.ExecuteNonQuery()
                conInd.Close()
            ElseIf int = 0 Then
                If txtDNI.TextLength <> 9 Then
                    MsgBox("Introduzca un DNI válido.", vbExclamation, "¡Error!")
                    txtDNI.Text = ""
                    txtDNI.Focus()
                    Return
                Else
                    For i = 0 To (txtDNI.MaxLength() - 1)
                        If txtDNI.Text(8) = "1" Or txtDNI.Text(8) = "2" Or txtDNI.Text(8) =
"3" Or txtDNI.Text(8) = "4" Or txtDNI.Text(8) = "5" Or txtDNI.Text(8) = "6" Or
txtDNI.Text(8) = "7" Or txtDNI.Text(8) = "8" Or txtDNI.Text(8) = "9" Or
txtDNI.Text(8) = "0" Then
                            MsgBox("El último carácter del DNI debe ser una letra.",
vbExclamation, "¡Error!")
                            txtDNI.Text = ""
                            txtDNI.Focus()
                            Return
                        ElseIf txtDNI.Text(i) = " " Then
                            MsgBox("No se puede introducir espacios en blanco.",
vbExclamation, "¡Error!")
                            txtDNI.Text = ""
                            txtDNI.Focus()
                            Return
                        ElseIf txtDNI.Text(i) = "º" Or txtDNI.Text(i) = "ª" Or txtDNI.Text(i)
= "\" Or txtDNI.Text(i) = "!" Or txtDNI.Text(i) = "|" Or txtDNI.Text(i) = "@" Or
txtDNI.Text(i) = "·" Or txtDNI.Text(i) = "#" Or txtDNI.Text(i) = "$" Or
txtDNI.Text(i) = "%" Or txtDNI.Text(i) = "&" Or txtDNI.Text(i) = "¬" Or
txtDNI.Text(i) = "/" Or txtDNI.Text(i) = "(" Or txtDNI.Text(i) = ")" Or txtDNI.Text(i)
= "=" Or txtDNI.Text(i) = "'" Or txtDNI.Text(i) = "?" Or txtDNI.Text(i) = "¡" Or
txtDNI.Text(i) = "¿" Or txtDNI.Text(i) = "€" Or txtDNI.Text(i) = "`" Or
txtDNI.Text(i) = "^" Or txtDNI.Text(i) = "[" Or txtDNI.Text(i) = "+" Or
txtDNI.Text(i) = "*" Or txtDNI.Text(i) = "]" Or txtDNI.Text(i) = "´" Or txtDNI.Text(i)
= "¨" Or txtDNI.Text(i) = "{" Or txtDNI.Text(i) = "Ç" Or txtDNI.Text(i) = "ç" Or
txtDNI.Text(i) = "}" Or txtDNI.Text(i) = "<" Or txtDNI.Text(i) = ">" Or
txtDNI.Text(i) = "," Or txtDNI.Text(i) = ";" Or txtDNI.Text(i) = "." Or txtDNI.Text(i)
= ":" Or txtDNI.Text(i) = "-" Or txtDNI.Text(i) = "_" Or txtDNI.Text(i) = "'" Then
```

```
                    MsgBox("Solo puede introducirse números y en la última
posición una letra.", vbExclamation, "¡Error!")
                    txtDNI.Text = ""
                    txtDNI.Focus()
                    Return
                End If
            Next
            MsgBox("El DNI Introducido no se encuentra en la Base de
Datos.", vbExclamation, "¡Error!")
            txtDNI.Text = ""
            txtDNI.Focus()
            Return
        End If
    End If
ElseIf intModificar = True Then
    Dim j As Integer
    j = dgvListado.CurrentCell.RowIndex
    conInd.Open()
    Dim cmd As New OleDb.OleDbCommand("UPDATE " & frmBase.ususql
& "_listado " & "SET [Check] = False", conInd)
    cmd.CommandType = CommandType.Text
    cmd.ExecuteNonQuery()
    conInd.Close()
    actualizar_tabla()
End If
cmdAgregar.Enabled = True
cmdEliminar.Enabled = True
cmdModificar.Enabled = True
cmdGuardar.Enabled = False
cmdCancelar.Enabled = False
dgvListado.ReadOnly = True
intEliminar = False
intModificar = False
actualizar_tabla()
End Sub

Private Sub cmdCancelar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdCancelar.Click
    cmdAgregar.Enabled = True
    cmdEliminar.Enabled = True
    cmdModificar.Enabled = True
    cmdGuardar.Enabled = False
    cmdCancelar.Enabled = False
    dgvListado.ReadOnly = True
    intEliminar = False
    intModificar = False
    actualizar_tabla()
End Sub
'-----------------------------------------------------------------------------------------------
```

```
    Private Sub dgvListado_CellEndEdit(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellEndEdit
    End Sub

    Private Sub dgvListado_CellValueChanged(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellValueChanged
    End Sub

    Private Sub dgvListado_CellClick(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellClick
    End Sub

    Private Sub dgvListado_CellContentClick(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellContentClick
        Dim j As Integer = dgvListado.CurrentRow.Index
        Dim upd As New OleDb.OleDbCommand("UPDATE " & frmBase.ususql &
"_listado SET [Check] = " & dgvListado.Item("Informe",
j).EditedFormattedValue.ToString & " WHERE [ID] = " & j + 1 & "", conInd)
        conInd.Open()
        upd.CommandType = CommandType.Text
        upd.ExecuteNonQuery()
        conInd.Close()
        Dim ID As String = "SELECT DNI FROM " & frmBase.ususql & "_listado" &
(Chr(13)) & "WHERE Check = True"
        Dim DA As New OleDbDataAdapter(ID, conInd)
        Dim DT As New DataTable
        Dim DNI As String = ""
        DA.Fill(DT)
        For Each dr As DataRow In DT.Rows
            DNI = dr.Item(0).ToString
            If DNI <> "" Then
                Me.Hide()
                frmInforme.Show()
                limpiar()
                Dim DAI As New OleDbDataAdapter("SELECT [Empleo], [Nombre],
[Primer Apellido], [Segundo Apellido], [DNI], [Sexo], [Fecha de Nacimiento],
[Teléfono], [Correo Electrónico], [Pn], [Antigüedad en el Ejército], [Antigüedad
de Empleo] FROM " & frmBase.ususql & "_listado WHERE DNI = '" & DNI & "'",
conInd)
                Dim DSI As New DataSet
                DAI.Fill(DSI, frmBase.ususql & "_listado")
                frmInforme.dgvDatos.DataSource = DSI.Tables(frmBase.ususql &
"_listado")
                Dim DAH As New OleDbDataAdapter("SELECT * FROM " & DNI &
"_informe", conInd)
                Dim DSH As New DataSet
```

```
            DAH.Fill(DSH, DNI & "_informe")
            frmInforme.dgvListado.DataSource = DSH.Tables(DNI & "_informe")
            frmInforme.dgvDatos.ReadOnly = True
            frmInforme.dgvListado.Columns.Item("ID").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Fecha Inicio
Actividad").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Fecha Fin
Actividad").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Actividad").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Duración").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Lugar").ReadOnly = True
            frmInforme.dgvListado.Columns.Item("Calificación").ReadOnly =
False
        End If
    Next
  End Sub
End Class
```

10. Formulario agregar personal

```
Imports System.Data.OleDb
Public Class frmAgregar
  Private Sub frmAgregar_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
    dtpNac.Value = "#01/01/" & Today.Year - 18 & "#"
    dtpAntEje.Value = Today
    dtpAntEmp.Value = Today
    cmbEmpleo.Select()
  End Sub

  Private Sub cmdAceptar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdAceptar.Click
    Dim con As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")
    Dim DNI As String = "SELECT DNI FROM " & frmBase.ususql & "_listado "
& "WHERE DNI = '" & txtDNI.Text & "'"
    Dim int As Integer = 0
    Dim identidad As String = ""
    Try
      Dim DAD As New OleDbDataAdapter(DNI, con)
      Dim DTD As New DataTable
      DAD.Fill(DTD)
      For Each drd As DataRow In DTD.Rows
        identidad &= drd.Item(0).ToString
        int = int + 1
      Next
    Catch ex As Exception
      MsgBox(ex.Message)
```

```
        End Try

        If int = 0 Then
            con.Open()
            Try
                Dim cmd As New OleDb.OleDbCommand("INSERT INTO " &
frmBase.ususql & "_listado VALUES (@ID, @Check, @Empleo, @Nombre,
@Apellido1, @Apellido2, @DNI, @Sexo, @FechaNac, @Tfno, @Pn,
@correo_electronico, @AntEje, @AntEmp)", con)
                cmd.CommandType = CommandType.Text
                Dim sel As String = "SELECT COUNT (ID) FROM " & frmBase.ususql
& "_listado"
                Dim DA As New OleDbDataAdapter(sel, con)
                Dim DT As New DataTable
                Dim count As Integer = 0
                DA.Fill(DT)
                For Each dr As DataRow In DT.Rows
                    count = CInt(dr.Item(0).ToString) + 1
                Next
                cmd.Parameters.Add(New OleDb.OleDbParameter("@ID",
OleDb.OleDbType.Integer))
                cmd.Parameters("@ID").Value = count
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Check",
OleDb.OleDbType.Boolean))
                cmd.Parameters("@Check").Value = "False"
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Empleo",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Empleo").Value = cmbEmpleo.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Nombre",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Nombre").Value = txtNombre.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido1",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Apellido1").Value = txtApellido1.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Apellido2",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Apellido2").Value = txtApellido2.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@DNI",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@DNI").Value = txtDNI.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Sexo",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Sexo").Value = cmbSexo.Text
                cmd.Parameters.Add(New OleDb.OleDbParameter("@FechaNac",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@FechaNac").Value = dtpNac.Value.ToString
                cmd.Parameters.Add(New OleDb.OleDbParameter("@Tfno",
OleDb.OleDbType.VarChar))
                cmd.Parameters("@Tfno").Value = txtTfno.Text
```

```
            cmd.Parameters.Add(New OleDb.OleDbParameter("@Pn",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@Pn").Value = txtPn.Text
            cmd.Parameters.Add(New
OleDb.OleDbParameter("@correo_electronico", OleDb.OleDbType.VarChar))
            cmd.Parameters("@correo_electronico").Value = txtCorreo.Text
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEje",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEje").Value = dtpAntEje.Value.ToString
            cmd.Parameters.Add(New OleDb.OleDbParameter("@AntEmp",
OleDb.OleDbType.VarChar))
            cmd.Parameters("@AntEmp").Value = dtpAntEmp.Value.ToString
            cmd.ExecuteNonQuery()
            Dim cre As New OleDb.OleDbCommand("CREATE TABLE " &
txtDNI.Text & "_informe(" & (Chr(13)) & "ID INT," & (Chr(13)) & "[Fecha Inicio
Actividad] DATETIME," & (Chr(13)) & "[Fecha Fin Actividad] DATETIME," &
(Chr(13)) & "Actividad VARCHAR (50)," & (Chr(13)) & "Duración VARCHAR
(50)," & (Chr(13)) & "Lugar VARCHAR (50)," & (Chr(13)) & "Calificación INT," &
(Chr(13)) & "PRIMARY KEY (ID))", con)
            cre.ExecuteNonQuery()
        Catch ex As Exception
            MsgBox("Se ha producido un error al introducir los datos." &
(Chr(13)) & "Vuelva a intentarlo.", vbExclamation, "¡Error!")
        End Try
        con.Close()
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Quiere realizar algún otro registro?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            txtNombre.Text = ""
            txtApellido1.Text = ""
            txtApellido2.Text = ""
            txtDNI.Text = ""
            txtTfno.Text = ""
            txtPn.Text = ""
            txtCorreo.Text = ""
            cmbEmpleo.Select()
            Return
        Else
            Me.Close()
            frmListado.actualizar_tabla()
            frmListado.Show()
        End If
    Else
        MsgBox("El DNI ya figura en la Base de Datos.", vbExclamation,
"¡Error!")
    End If
End Sub
```

```
    Private Sub cmdCancelar_Click(sender As System.Object, e As
System.EventArgs) Handles cmdCancelar.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de que quiere cancelar?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmListado.actualizar_tabla()
            frmListado.Show()
        End If
    End Sub

    Private Sub dtpNac_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpNac.ValueChanged
        dtpAntEje.Select()
    End Sub

    Private Sub dtpAntEje_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpAntEje.ValueChanged
        dtpAntEmp.Select()
    End Sub

    Private Sub dtpAntEmp_ValueChanged(sender As System.Object, e As
System.EventArgs) Handles dtpAntEmp.ValueChanged
        Me.Select()
    End Sub
End Class
```

11. Formulario informe personal

```
Public Class frmInforme
    Dim conInd As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")

    Private Sub frmInforme_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
    End Sub

    Private Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        frmListado.Show()
        frmListado.actualizar_tabla()
        Me.Close()
    End Sub
```

```
    Private Sub dgvListado_CellValueChanged(sender As System.Object, e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvListado.CellValueChanged
        Dim j As Integer = dgvListado.CurrentRow.Index
        Dim act As New OleDb.OleDbCommand("UPDATE " &
dgvDatos.Item("DNI", 0).Value.ToString & "_informe SET [Calificación] = " &
dgvListado.Item("Calificación", j).EditedFormattedValue.ToString & "", conInd)
        conInd.Open()
        act.CommandType = CommandType.Text
        act.ExecuteNonQuery()
        conInd.Close()
    End Sub
End Class
```

12. Formulario histórico de actividades

```
Imports System.Data.OleDb

Public Class frmHistorico
    Dim conCom As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_comunes.accdb")
    Dim conInd As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=..\tablas_individualizadas.accdb")
    Private Sub frmHistorico_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        actualizar_tabla()
        txtIni.Text = frmBase.usuario
    End Sub

    Public Sub actualizar_tabla()
        Dim DAH As New OleDbDataAdapter("SELECT * FROM " &
frmBase.ususql & "_historico ORDER BY [Fecha Inicio Actividad]", conInd)
        Dim DSH As New DataSet
        DAH.Fill(DSH, frmBase.ususql & "_historico")
        dgvListado.DataSource = DSH.Tables(frmBase.ususql & "_historico")
    End Sub

    Private Sub cmdVolver_Click(sender As System.Object, e As
System.EventArgs) Handles cmdVolver.Click
        Dim dlgCerrar As Integer
        dlgCerrar = MsgBox("¿Está seguro de desea volver al Escritorio?", _
        vbYesNo + vbQuestion, "Mensaje")
        If dlgCerrar = vbYes Then
            Me.Close()
            frmEscritorio.Show()
        End If
    End Sub
End Class
```