UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DISEÑO Y GESTIÓN DE UNA BASE DE DATOS DE USUARIOS TEMPORALES EN EL DEPARTAMENTO DE ORGANIZACIÓN DE EMPRESAS UPV

Jorge Zafrilla Muñoz

Tutor: **Fernando González Ladrón de Guevara**

Cotutores**: Francisco J. Villar Villanueva, Raúl Blasco Herrera**

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia, 9 de septiembre de 2018

Resumen

El Departamento de Organización de Empresas de la Universitat Politècnica de València necesita poder registrar y controlar el personal externo que de forma temporal mantiene una relación con el mismo. Como consecuencia de esta relación hay que activar y proporcionar a estos usuarios una serie de recursos o servicios informáticos que proporciona el departamento.

El centro cuenta actualmente con tres tipos de documentos, cada uno para un tipo de perfil de usuario temporal, que se rellenan a mano. Al realizarse en papel y de forma manual no existe ningún control sobre fechas de caducidad, tampoco automatismos para dar de baja o alta a los usuarios temporales en los distintos registros o sistemas software que van a utilizar.

En este TFG se encuentra una descripción de mi programa realizado en Python que consiste en una interfaz gráfica de fácil manejo para el usuario mediante la cual se gestiona una Base de datos que he creado en SQLite.

Además de una completa información acerca del lenguaje de programación Python, y del sistema de gestión de bases de datos de SQLite y un manual de usuario para la correcta utilización del programa desarrollado.

Resum

El Departament d'Organització d'Empreses de la Universitat Politècnica de València necessita poder registrar i controlar el personal extern que de forma temporal manté una relació amb el mateix. Com a conseqüència d'aquesta relació cal activar i proporcionar a estos usuaris una sèrie de recursos o serveis informàtics que proporciona el departament.

El centre compta actualment amb tres tipus de documents, cada un per a un tipus de perfil d'usuari temporal, que s'omplin a mà. Al realitzar-se en paper i de forma manual no hi ha cap control sobre les dates de caducitat, tampoc automatismes per a donar de baixa o d' alta als usuaris temporals en els diferents registres o sistemes software que utilitzaran.

En este TFG trobaràs una descripció del meu programa realitzat en Python que consistix en una interfície gràfica fàcil de manejar per l'usuari per mitjà de la qual es gestiona una base de dades que he creat en SQLite.

A més d'una completa informació sobre el llenguatge de programació Python, i del sistema de gestió de bases de dades de SQLite i un manual d'usuari per a la correcta utilització del programa desenvolupat.

Abstract

The Department of Business Organization of the Universitat Politècnica de València needs to be able to register and control external personnel who temporarily maintain a relationship with it. As a consequence of this relationship, it is necessary to activate and provide these users with a series of resources or computer services provided by the department.

The center currently has three types of documents, each for a type of temporary user profile, which are filled in by hand. Due to be performed on paper and manually there is no control over expiration dates, nor are there automatisms to unsubscribe or discharge temporary users in the different registers or software systems that they will use.

In this TFG there is a description of my program made in Python that consists of a user-friendly graphic interface for the user through which a database that I created in SQLite is managed.

In addition to a complete information about the programming language Python, and the database management system of SQLite and a user manual for the correct use of the developed program.

Índice

Índice de figuras

Índice de tablas

1. Introducción

El proyecto que voy a describir a continuación basa sobre el desarrollo de un programa para gestionar una Base de datos destinada a la gestión de usuarios temporales del Departamento de Organización de Empresas de la Universitat Politècnica de València.

El Departamento necesita poder registrar y controlar el personal externo, que de forma temporal mantiene una relación con el mismo. Como consecuencia de esta relación hay que activar y proporcionar a estos usuarios una serie de recursos o servicios informáticos que proporciona el departamento [1].

1.1. Departamento de Organización de Empresas

El Departamento de Organización de Empresas de la Universitat Politècnica de València incluye a los profesores adscritos al área de conocimiento de "Organización de Empresas". El Departamento cuenta con más de 140 profesores e imparte su docencia en 10 de los 14 centros de la Universidad, participando en la docencia de, prácticamente, todos los Grados y en más de 11 programas de másteres universitarios.

Los profesores del Departamento son los responsables de más del 5% de la docencia que se imparte en la UPV, y la valoración que la UPV hace de la Actividad Docente e Investigadora lo convierte en uno de los departamentos mejor calificados.

Los profesores del DOE dirigen más de 150 TFG y TFM por curso. De entre las tesis presentadas por miembros del Departamento en el curso pasado, dos fueron galardonadas con el selectivo "Premio Extraordinario de Doctorado" de la UPV.

Los usuarios temporales de dicho departamento son personal investigador y docente visitantes (profesores, becarios, etc.) a los que hay que proporcionar una serie de recursos [2].

1.2. Situación inicial y requirimientos

El departamento cuenta actualmente con tres tipos de documentos (Figura 1), cada uno para un tipo de perfil de usuario temporal, que se rellenan a mano con el fin de tomar datos y registrar estos colaboradores temporales. Estos documentos de alta están disponibles de forma pública en la web del departamento [7][8][9].

Al realizarse en papel y de forma manual no existe ningún control sobre fechas de caducidad, tampoco automatismos para dar de baja o alta a los usuarios temporales en los distintos registros o sistemas software que se van a utilizar. Además en muchas ocasiones, no se sigue un procedimiento claro en cuanto a cuándo y quién debe rellenar el documento de alta.

Por último, también existe una hoja de cálculo Excel donde se almacena la información recogida por los citados documentos en papel, en la que se ha implementado una macro, que avisa acerca de usuarios que en los próximos 15 días deben darse de baja de los sistemas [1].

La hoja de cálculo Excel es la que va a ser sustituida por la Base de datos. Se compone de dos hojas, una para usuarios activos y otra para inactivos, al igual que las tablas que tendrá la Base de datos. Los campos de dicho Excel son los mismos que va a tener la Base de datos (Tabla 1).

Usuarios	Nombre y apellidos	Email	Teléfono
Universidad de origen	Ciudad	País	Universidad Profesor asociado
Profesor anfitrión DOE	Email profesor	Fecha llegada	Fecha salida prevista
Fecha salida real	Ubicación DOE	Tipo ordenador	Alfresco
Informacion adicional	Usuario UPVNET o identificación	Asignar usuario temporal	AD Imprimir
Papercut	Activado en fotocopiadora	Código	Activar fotocopiadora en piso
Comedor	Despachos asociados	Seminario Informática	Seminario Teleco
Seminario 3.1	Seminario 3.2	Seminario 3.3	Seminario 4
Actualizador	Laboratorios	Taquillas	Otros, indicar

Tabla 1. Campos de la hoja de cálculo y de la base de datos.

doe
Departamento de Organización de Empresas

ALTA USUARIO PROFESOR VISITANTE

Fecha de Alta ___ /___ /___

Nombre y Apellidos usuario:		
Dirección de e-mail:		
Profesor responsable:		
Periodo estancia:	Desde: ___ /___ /___	Hasta: ___ /___ /___
Tiene usuario UPVNET:	SI ☐_____	NO ☐ Indicar doc. identificación: _____
Asignar usuario temporal:	SI ☐	NO ☐
Ordenador personal:	PROPIO: ☐	Solicita a DOE: ☐

Tarjeta acceso puertas DOE[1]:

Comedor ☐	**Dpchos. Asociados ☐**	**Seminario 4 ☐**	**Otro, especificar☐** _____
Actualizador☐			

OBSERVACIONES:

Para cualquier información sobre servicios DOE-TIC consultar la página web del DOE www.doe.upv.es, menú de "Recursos".

Figura 1. Formulario actual en papel para usuario Profesor visitante [9].

2. Descripción general

En esta parte de la memoria se encuentra el objetivo del trabajo, cuáles son los propósitos que se desean conseguir. Los requisitos que deberá tener para que el programa sirva de utilidad para el departamento, ya que nos gustaría que el programa se integrara en la página web actual del departamento para que quedara operativo. También incluye una descripción del software utilizado para el proyecto.

2.1. Objetivo del trabajo

La motivación del proyecto es implementar un programa para gestionar una base de datos para usuarios y los servicios asignados a los mismos. Debe ser desarrollada mediante software libre y ser accesible a usuarios registrados que dispongan de una conexión a Internet y un navegador instalado en su ordenador. El programa debe de servir de utilidad para el departamento, ya que queremos que se integre en la página web actual del mismo para que quede operativo.

Este programa debe contar con una interfaz que permita al operador introducir los datos relativos a cada tipo de usuario y éstos serán incluidos de manera automática en la base de datos creada. Además debe tener una serie de menús, en los que el operador pueda editar cualquier campo y también eliminar registros. Todo esto, de manera suficientemente intuitiva para el operador.

2.2. Requisitos

- Deberá estar desarrollado íntegramente con software libre, de tal manera que no sea necesario el desembolso de cantidad económica alguna, ni para su puesta en marcha ni posterior mantenimiento. El lenguaje de programación escogido será Python.
- Dado el reducido tamaño o volumen de datos a manejar, la Base de datos deberá estar implementada con SQLite [1].

2.3. Software utilizado

Para este trabajo es necesario un software adecuado:

- Un lenguaje de programación con el que implementar el código.
- Una interfaz de usuario para que programar en el lenguaje correspondiente sea cómodo.
- Un lenguaje de acceso a BBDD relacionales con el que poder manipular los elementos de la base de datos.

En la tabla 2 se muestra el software finalmente escogido para cada necesidad.

Python	**Lenguaje de programación**
Eclipse	Interfaz de usuario
SQLite	Lenguaje de acceso a BBDD

Tabla 2. Software escogido para el trabajo.

2.3.1. Python

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con el tipado dinámico y la vinculación dinámica, que significa que el programa no conoce el tipo de las variables (int, String...) ni el código que se ejecutará para una llamada de procedimiento específica, hasta el momento de la ejecución, ambas lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como también para usarlo como scripting o lenguaje de pegado para conectar componentes existentes.

La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario sin cargo para todas las plataformas principales, y se pueden distribuir libremente.

Como no existe un paso de compilación, el ciclo de edición-prueba-depuración es increíblemente rápido. La depuración de programas Python es fácil: un error o entrada incorrecta nunca causará un error de segmentación. En cambio, cuando el intérprete descubre un error, genera una excepción. Cuando el programa no capta la excepción, el intérprete imprime un seguimiento de la pila. Un depurador de nivel de fuente permite la inspección de variables locales y globales, la evaluación de expresiones arbitrarias, el establecimiento de puntos de interrupción, recorrer el código una línea a la vez, y así sucesivamente. El depurador está escrito en Python, lo que demuestra el poder introspectivo de Python [3].

Hemos elegido este lenguaje para el proyecto por su facilidad de uso. La estructura del código es bastante natural y promueve una forma de escribir que facilita su lectura. Además de que es un lenguaje de código abierto, nos decantamos por este lenguaje en vez de otros como por ejemplo Java, porque Python es totalmente libre, mientras que Java, que aunque en sí mismo sea libre, puede estar limitado por alguna dependencia de software que no es libre. Este hecho hace que no haya gasto económico alguno. Para este proyecto he utilizado la versión 3.6.

Figura 2. Logo de Python.

2.3.2. Eclipse

Eclipse es una plataforma que se ha diseñado para la creación de herramientas integradas de desarrollo web y de aplicaciones.

Eclipse proporciona un modelo de interfaz de usuario común para trabajar con herramientas. Está diseñado para ejecutarse en múltiples sistemas operativos, a la vez que brinda una sólida integración con cada sistema operativo subyacente. Los complementos pueden programarse en las API portátiles de Eclipse y ejecutarse sin cambios en cualquiera de los sistemas operativos admitidos.

En el núcleo de Eclipse se encuentra una arquitectura para descubrimiento dinámico, carga y ejecución de complementos. La plataforma maneja la logística de encontrar y ejecutar el código correcto. La interfaz de usuario de la plataforma proporciona un modelo de navegación de usuario estándar, cada complemento puede enfocarse en hacer bien una pequeña cantidad de tareas (definir, probar, compilar, depurar, etc.) [4].

He utilizado la interfaz de usuario Eclipse para una mayor organización del código y facilidad de ejecución, existen otras como Pycharm o Wing, pero decidí utilizar Eclipse por su compatibilidad con gran variedad de lenguajes, por lo que aprender su funcionamiento puede servirme para futuros trabajos con otros lenguajes. La versión que he utilizado es Eclipse Photon.

Figura 3. Logo de la versión utilizada de Eclipse, versión de junio de 2018.

2.3.3. SQL y SQLite

SQL es un lenguaje estándar e interactivo de acceso a BBDD relacionales que permite especificar diversos tipos de operaciones en ellas, gracias a la utilización del álgebra y de cálculos relacionales. SQL brinda la posibilidad de realizar consultas con el objetivo de introducir, recuperar, editar y eliminar información de las bases de datos de manera sencilla. Las consultas toman la forma de un lenguaje de comandos que permite seleccionar, insertar, actualizar, averiguar la ubicación de los datos, y borrarlos [5].

SQLite es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional independiente, sin servidor y de configuración cero. El código para SQLite es de dominio público y, por lo tanto, es gratuito para cualquier uso, comercial o privado. SQLite es la base de datos más implementada del mundo con más aplicaciones de las que podemos contar, incluidos varios proyectos de alto perfil.

A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor por separado. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con múltiples tablas, índices, desencadenadores y vistas. Está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits o entre arquitecturas *big-endian* y *little-endian*. Estas características hacen que SQLite sea una opción popular como formato de archivo de aplicación. Los archivos de base de datos SQLite son un formato de almacenamiento recomendado por la Biblioteca del Congreso de EE.UU. [6].

Para este proyecto se ha realizado la importación en Python de la biblioteca SQLite3 para la gestión de la base de datos, ya que no requiere configuración ni uso de un servidor.

Figura 4. Logo SQLite.

3. Metodología

La metodología empleada consiste, para comenzar, en conocer que es lo que el trabajo demanda. Una vez comprendido y asimilado que es lo que queremos, por medio de tareas, se gestiona la manera de llevar a cabo este proyecto para su correcta realización.

3.1. Gestión del proyecto

Primeramente hay que conocer lo que demandan los usuarios. Actualmente en el Departamento de Organización de Empresas se utilizan en formato papel tres tipos de documentos de alta, que se rellenan a mano, uno para cada tipo de usuario, posteriormente los datos recogidos se pasan a una hoja de Excel.

La función de este programa es obtener dichos formularios en una interfaz gráfica en la que el operador puede hacer click en el tipo de documento de alta que quiere rellenar, en ese momento aparecerán por pantalla los campos que tiene cada tipo de usuario para rellenar. Además de los recursos informáticos de los que puede disponer, el operador podrá hacer click en cada uno de los recursos para seleccionar o deseleccionar.

Una vez introducidos los datos y seleccionados los recursos que se desean activar, se pulsará un botón que introducirá todos estos datos en la base de datos, en la tabla correspondiente y en sus respectivos campos.

También contará con una barra de menú en la que se podrá seleccionar la opción de editar campos, donde se seleccionará el nombre del usuario del que se quiere editar un campo, después se elige el que se desee editar y se sustituye en la base de datos.

Otra opción de la barra de menú será la de eliminar un registro, aquí se seleccionará el nombre del usuario que se quiere eliminar y se borrará completamente su registro.

Contará además con un algoritmo que calcule, mediante las fechas de salida introducidas, qué usuarios deben ser dados de baja en menos de 15 días. En la barra de menú habrá un botón que al pulsarlo aparecerá una lista con los nombres de los usuarios que deben de ser dados de baja en ese tiempo. También cuenta con un botón para salir del programa.

3.2. Tareas

Las tareas a realizar serán en primer lugar aprender a utilizar el lenguaje necesario, Python. Una vez conocido, habrá que investigar acerca de sus herramientas para crear interfaces gráficas, también aprender a utilizar SQLite, para una vez creada la interfaz gráfica, la base de datos pueda ser creada y gestionada.

3.2.1. Aprendizaje de Python

El primer paso a la hora de empezar con el trabajo fue buscar información sobre el programa que iba a emplear, ya que nunca antes lo había utilizado. Para ello consulté una colección de videos en YouTube acerca de la programación en Python desde cero [10]. Este lenguaje resultó ser muy similar al lenguaje Java, el cuál he aprendido en asignaturas del Grado como Programación y Diseños de Servicios Telemáticos, por lo que me sirvieron de mucha ayuda los conocimientos de programación adquiridos en la carrera.

3.2.2. Desarrollo de la interfaz gráfica

Una vez aprendido las bases del lenguaje de programación Python, quise aprender a crear una interfaz gráfica con éste, así que recurrí otra vez a una colección de videos en YouTube [11] que explicaban desde nivel cero como crear pequeñas aplicaciones gráficas empleando la librería de Python llamada Tkinter de la cual hablaré más adelante cuando explique el desarrollo del trabajo.

Una vez realizada la tarea de aprender a crear aplicaciones gráficas, llegó el momento de empezar a emplear en mi programa el conocimiento adquirido. En primer lugar fue la implementación de cada uno de los perfiles de usuario con los que cuenta el DOE: doctorando, profesor asociado y profesor visitante.

Una vez creados todos los campos necesarios para cada usuario, el siguiente paso era obtener los datos introducidos por el operador para poder trabajar con ellos. Con los datos necesarios, la siguiente tarea consiste en trabajar con ellos en la base de datos.

Finalmente creo unos botones en la barra de menú, en los que se puede editar los campos de la base de datos y eliminar registros. Además de una lista con los nombres de los usuarios cuya estancia está a punto de caducar y un botón de salida del programa.

3.2.3. Implementación de la base de datos

Para esta tarea, como ya hice en las anteriores, lo primero de todo fue ver una serie de videos en YouTube para aprender cómo crear una base de datos y gestionarla en Python [11]. Para tal objetivo utilizo la librería SQLite, en mi programa importo la librería sqlite3. Con el lenguaje ya aprendido, es hora de emplearlo en mi programa.

Lo primero fue crear la base de datos, una vez creada le introduje dos tablas, una con los usuarios activos y otra con los inactivos. Es en el desarrollo de la tabla donde se introducen los campos que van a tener cada una (nombre y apellidos, teléfono, etc.) obtenidos mediante la aplicación gráfica, en la que el operador ha introducido todos estos datos.

Una vez creadas las tablas y los campos, mediante comandos de SQLite, éstos pueden ser modificados, a la vez que se pueden eliminar registros.

4. Desarrollo y resultados del trabajo

A la hora de desarrollar este trabajo, lo primero que voy a hacer es explicar en qué consiste una interfaz gráfica con la librería Tkinter y sus distintas partes. Una vez conocido el funcionamiento y la utilidad de éstos, procederé a explicar las distintas funciones que contiene el formulario, en la que se emplean los objetos de dicha librería.

Una vez creadas las funciones ya se pueden utilizar para la creación de los distintos perfiles de usuario, cada uno con sus correspondientes campos.

Con la interfaz principal completa, ya se pueden obtener los datos y trabajar con ellos en la base de datos, para ello se utiliza SQLite, cuyos comandos explicaré más adelante.

Ya con los datos conseguidos e introducidos en la base de datos, crearé funciones para su posible edición y una interfaz para poder manipularla de una forma intuitiva.

4.1. Librería Tkinter

Tkinter es una adaptación de la biblioteca Tcl/Tk para el lenguaje de programación Python y otros lenguajes como Perl y Rubí [13].

Una interfaz gráfica consta de tres elementos principales: ventana, frame y widget (Figura 5).

4.1.1.Elementos de una interfaz gráfica

- Ventana: es el área visual principal de la interfaz, se introducen en ella los frames y los widgets. Para que la ventana sea lanzada, al final de la interfaz debe aparecer el método "ventana.mainloop()".
- Frame: divide el área de la ventana en diferentes partes independientes unas de otras, sirve para aglutinar los widgets dentro. Su utilización no es obligatoria, los widgets podrían estar directamente dentro de la ventana, pero es recomendable usar el frame para mantener un orden. El frame también es un widget.
- Widget: son los elementos dentro de la ventana o del frame, se conoce como widgets a: botones, entradas de texto, etiquetas, menús desplegables, etc.

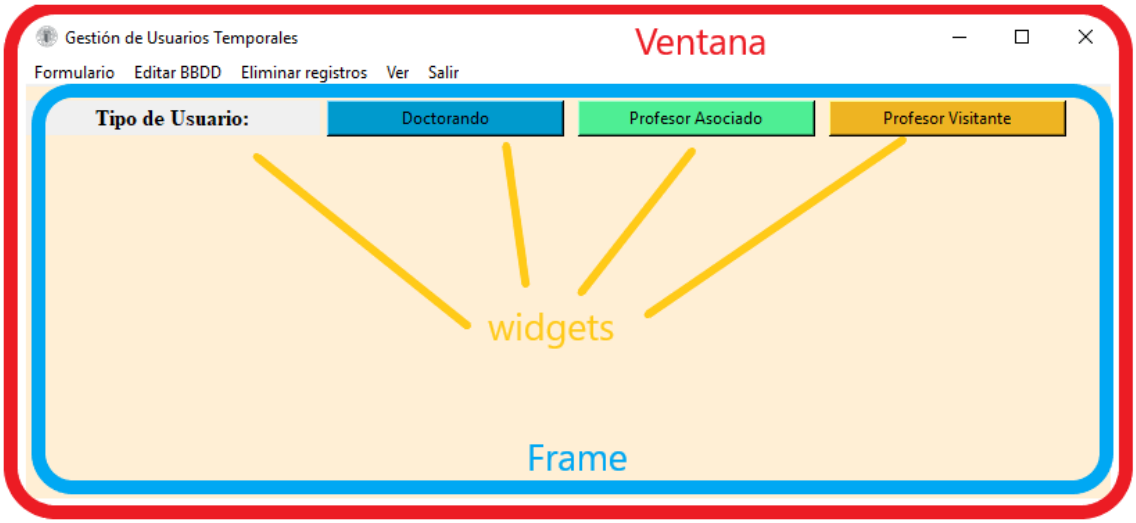

Figura 5.Esquema de cada elemento de la interfaz gráfica.

4.1.2. Widgets utilizados

Los widgets utilizados en este programa, ordenados de mayor a menor uso son los siguientes:

Label	Sirven para mostrar líneas de texto, mapas de bits o imágenes.
Entry	Permite que el usuario vea y modifique una sola línea de texto. En este widget es en el que el usuario introduce los datos por teclado.
Checkbutton	Crea una casilla de selección para que el usuario lea y seleccione una opción de las dos posibles.
Radiobutton	Sirven para que el usuario seleccione sólo una del conjunto de opciones disponibles.
Button	Origina un botón. Una vez creado se le puede asignar una función para que se ejecute al pulsar el botón.
ttk.Combobox	Crea una lista desplegable.
Menu	Inserta una barra de menú en la ventana.
Frame	Divide la ventana y sirve para aglutinar widgets dentro.

Tabla 3. Widgets utilizados en el programa. Fuente: Tkinter 8.5 reference: a GUI for Python [12].

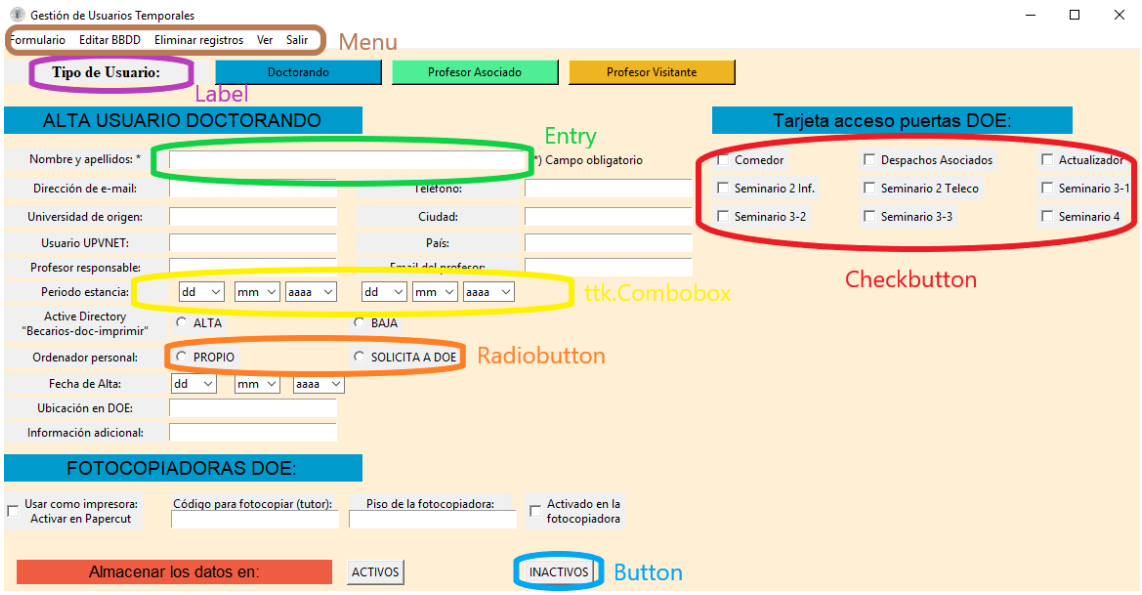

Figura 6.Widgets utilizados en la interfaz gráfica.

Todo widget tiene argumentos, que son los que dirigen el comportamiento y apariencia del widget, dentro de éstos, el primero e imprescindible es lo que se denomina como parent, éste es el espacio donde está incluido dicho widget, puede ser directamente en la ventana o dentro de un frame. Después del parent los siguientes argumentos son opciones que tiene cada widget.

width	Define el ancho en caracteres, cuando se trata de texto, o en píxeles, cuando es una imagen.
height	Altura en caracteres, en texto, o píxeles, en imágenes.
bg o background	Color de fondo.
text	Texto que mostrará dicho widget.
textvariable	El texto que mostrará el widget es una variable e irá cambiando.
font	Cuando hay texto, sirve para especificar la fuente.
justify	Especifica el alineado de las líneas de texto.
relief	Estilo del relieve.
variable	Define una variable de control.
value	Valor que se define en la variable de control cuando el botón es pulsado.
onvalue	En un Checkbutton, sirve para establecer el valor definido en la variable de control cuando el botón está pulsado.
offvalue	En un Checkbutton, sirve para establecer el valor definido en la variable de control cuando el botón no está pulsado.
state	En un ttk.Combobox establece el estado de la lista desplegable, es decir si se puede editar o no.
command	En un Button, se define una función a realizar cuando éste es pulsado.
cursor	Apariencia del cursor cuando éste está sobre el widget.
tearoff	En el widget Menu, las opciones se agregarán comenzando desde la posición indicada.

Tabla 4. Opciones de widget [12]

4.1.3. El método .grid()

Una vez conocidos y creados los widgets la siguiente cuestión es cómo colocarlos. Este método sirve para crear y administrar una geometría de cuadrícula para un widget, de esta forma trabajamos con filas y columnas para colocar cada elemento en su lugar.

Los argumentos utilizados de dicho método son:

column	El número de columna en el que se cuadra el widget, contando desde cero.
row	El número de fila en la que se cuadra el widget, contando desde cero.
sticky	Cuando un widget no ocupa totalmente el tamaño de su celda, esta opción sirve para distribuir dicho espacio sobrante.
padx	Número de espacios entre celdas a izquierda y derecha.
pady	Número de espacios entre celdas, arriba y abajo.
columnspan	Sirve para combinar columnas y hacer que ocupen el número de celdas que se especifique.

Tabla 5. Opciones del método .grid() [12].

También utilizo otro método para posicionar elementos, el método .pack(), en éste sólo especificamos si el elemento debe ir arriba, abajo, izquierda o derecha de un elemento de referencia o de la misma ventana.

Los argumentos utilizados de este método son:

fill	Determina si se debe utilizar más espacio o mantener las dimensiones propias.
expand	Establece si el widget se expandirá a la vez que el parent.

Tabla 6. Opciones del método .pack() [12]

4.2. Funciones del formulario

En este formulario, cada uno de los principales campos a rellenar de la interfaz gráfica, con los datos del usuario, han sido creados como funciones independientes, pero todas con una estructura que sigue el mismo patrón.

Conociendo el funcionamiento de los diferentes widgets de Tkinter es sencillo explicar las diferentes funciones.

4.2.1. Elementos de entrada por teclado

En esta aplicación para dar de alta a distintos perfiles de usuario, es necesario la introducción de los datos personales de éste, los cuales son introducidos por teclado por el operador, las funciones de este tipo siguen una misma estructura:

- Se declara la variable de control y la variable en la que se almacena el Entry de forma global, esto significa que se puede dar valor a dicha variable desde una subrutina y este valor será el de dicha variable para todos los niveles del programa.
- Se crea un Label en el que se introduce el texto deseado, en este caso se pide al usuario los datos que debe introducir en el Entry que tiene dicho Label a su derecha.
- Ahora se introduce el Entry, para que el usuario escriba los datos y éstos se impriman por pantalla. En el Entry se establece una variable de control, que será donde se va a almacenar la cadena de texto que el operador va a introducir, para luego trabajar con ella desde otras funciones, por eso se declara como variable global.
- En ambas se utiliza el método .grid() para colocar cada widget en su lugar de la cuadrícula.

global *variable, nombredelEntry*
nombredelLabel = Label(*parent*, text = *'Texto a mostrar por pantalla: '*, width = *número*)
nombredelLabel.grid (row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm)*
nombredelEntry = Entry(*parent*, textvariable = *variable*, justify = *'center'*, width = *núm*)
nombredelEntry.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, columnspan = *núm*)

Tabla 7. Estructura de funciones de entrada por teclado.

```
def nomape(num):
    labelNombre = Label(miFrame, text = 'Nombre y apellidos: *', width = 23)
    labelNombre.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = (0,2))
    global n,cuadroNombre
    cuadroNombre = Entry(miFrame,textvariable = n,justify = 'center',width = 60)
    cuadroNombre.grid(row = num, column = 1, sticky = 'w',columnspan = 2)
    labelCampob = Label(miFrame, text = '(*) Campo obligatorio ', bg = 'papaya whip')
    labelCampob.grid(row = num, column = 3, sticky = 'w', padx = 2, pady = 2)

def telEfono(num):
    labelTelefono = Label(miFrame, text = 'Teléfono:', width = 23)
    labelTelefono.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global tlf, cuadroTlf
    cuadroTlf = Entry(miFrame, textvariable = tlf,justify = 'center', width = 28)
    cuadroTlf.grid(row = num, column = 3, sticky = 'w')

def uniOrigen(num):
    labelorigen = Label(miFrame, text = 'Universidad de origen:', width = 23)
    labelorigen.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global uniorigen,cuadroorigen
    cuadroorigen = Entry(miFrame, textvariable = uniorigen,justify = 'center', width = 28)
    cuadroorigen.grid(row = num, column = 1, sticky = 'w')

def ciuDad(num):
    labelciudad = Label(miFrame, text = 'Ciudad:', width = 23)
    labelciudad.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global ciudad,cuadrociudad
    cuadrociudad = Entry(miFrame, textvariable = ciudad,justify = 'center', width = 28)
    cuadrociudad.grid(row = num, column = 3, sticky = 'w')
```

Figura 7. Algunas funciones para la introducción de datos por teclado.

4.2.2. Checkbuttons

Para dar de alta los distintos servicios de los que dispondrá el usuario, el programa lo realizará de una forma muy sencilla por medio de Checkbuttons, en los que al activar o desactivar las casillas, se dará de alta, o no, al servicio en cuestión. La estructura es la siguiente:

- Se declara como global la variable de control para poder darle valor en la subrutina y utilizar ese valor después en otras funciones fuera de dicha subrutina.
- En los argumentos del Checkbutton se introduce un texto que será el que aparecerá junto a la casilla de verificación, una variable en la que se almacenará el valor del botón, un onvalue en la que se introduce el valor que queremos que adquiera la variable cuando el botón está activado, y un offvalue con el valor cuando éste está desactivado.
- Como cada widget, utiliza el método .grid() para su colocación correspondiente.

global *variable*

Checkbutton(*parent*, text = *'Texto a mostrar junto al botón'*, variable = *variable*,

onvalue = *núm*, offvalue = *núm*).grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)

Tabla 8. Estructura de funciones con Checkbutton.

```
global comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,papercut,fotocopiadora,pisofoto,codfoto,codigofotocopiar,pisofotocopia
Checkbutton(miFrame, text = 'Comedor', variable = comedor,onvalue = 1,offvalue = 0).grid(row = 2, column = 4, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Despachos Asociados', variable = despachos,onvalue = 1,offvalue = 0).grid(row = 2, column = 5, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Actualizador', variable = actualizar,onvalue = 1,offvalue = 0).grid(row = 2, column = 6, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 2 Inf.', variable = seminf,onvalue = 1,offvalue = 0).grid(row = 3, column = 4, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 2 Teleco', variable = semtel,onvalue = 1,offvalue = 0).grid(row = 3, column = 5, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 3-1', variable = sem31,onvalue = 1,offvalue = 0).grid(row = 3, column = 6, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 3-2', variable = sem32,onvalue = 1,offvalue = 0).grid(row = 4, column = 4, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 3-3', variable = sem33,onvalue = 1,offvalue = 0).grid(row = 4, column = 5, sticky = 'w', padx = 20, pady = 2)
Checkbutton(miFrame, text = 'Seminario 4', variable = sem4,onvalue = 1,offvalue = 0).grid(row = 4, column = 6, sticky = 'w', padx = 20, pady = 2)
```

Figura 8. Creación de Checkbuttons para servicios de acceso del usuario Doctorando.

4.2.3. Radiobuttons

Cuando queramos escoger únicamente una opción dentro de varias posibles, utilizamos Radiobuttons, éstos son muy útiles en el formulario cuando queremos saber, por ejemplo, el centro en el que realiza la docencia el profesor visitante, o si el usuario tiene ordenador propio o tiene que proporcionárselo el DOE, entre otros.

Estructura:

- Al igual que las anteriores funciones, se declara como global la variable de control.
- Cuenta con un Label con el texto en el cual se solicitará al operador que seleccione una de las opciones disponibles.
- Se crean tantos Radiobutton como opciones se deseen, éstos deben tener en común la variable de control, para que adquiera un valor u otro dependiendo de la opción escogida.
- Todos los Radiobutton cuentan con un argumento value, en el que se especifica el valor que va a adoptar la variable, dependiendo del botón que pulse.
- Como siempre, cada widget utiliza la función .grid() para colocarlo en el lugar correspondiente.

global *variable*
nombredelLabel = Label(*parent*, text = *'Elige opción: '*, width = *núm*)
nombredelLabel.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)
Opcion1 = Radiobutton(*parent*, text = *'Opción 1'*, variable = *variable*, value = *núm*)
Opcion1.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)
Opcion2 = Radiobutton(*parent*, text = *'Opción 2'*, variable = *variable*, value = *núm*)
Opcion2.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)
Opcion3 = Radiobutton(*parent*, text = *'Opción 3'*, variable = *variable*, value = *núm*)
Opcion3.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)

Tabla 9. Estructura de funciones con Radiobutton.

```
def activeDirectory(num,col,x,y):
    global directory
    labelactivedir = Label(miFrame, text = 'Active Directory\n"Becarios-doc-imprimir"', width = 23)
    labelactivedir.grid(row = num, column = col, sticky = 'w', padx = x, pady = y)
    botonalta = Radiobutton(miFrame, text = 'ALTA', variable = directory, value = 1)
    botonalta.grid(row = num, column = col+1, sticky = 'w', padx = x, pady = y)
    botonbaja = Radiobutton(miFrame, text = 'BAJA', variable = directory, value = 2)
    botonbaja.grid(row = num, column = col+2, sticky = 'w', padx = x, pady = y)


#---------------------------Ordenador Personal--------------------

def ordenadorPersonal(num):
    global tienordenador
    labelordenador = Label(miFrame, text = 'Ordenador personal:', width = 23)
    labelordenador.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    propio = Radiobutton(miFrame, text = 'PROPIO', variable = tienordenador, value = 1)
    propio.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    doe = Radiobutton(miFrame, text = 'SOLICITA A DOE', variable = tienordenador, value = 2)
    doe.grid(row = num, column = 2, sticky = 'w', padx = 2, pady = 2)
```

Figura 9. Algunas funciones utilizadas para crear Radiobuttons.

4.2.4. Buttons

Un elemento conocido por todos, casi indispensable en cualquier programa son los botones. En este caso utilizados desde el primer momento de ejecución para elegir el tipo de perfil de usuario y desplegar el formulario adecuado. También para insertar, modificar y eliminar los datos de la base de datos. La estructura es como la que se muestra a continuación:

- En este caso no cuenta con una variable de control global como en las anteriores funciones. El botón en sí solamente es un elemento visual de la interfaz gráfica, es necesario el argumento command para asignarle una función a realizar cuando sea pulsado.
- La función designada con command será una función de Python, que estará definida previamente y específica para ese botón.
- Este widget también cuenta con el argumento relief, que elige el tipo de relieve del botón. El argumento cursor cambia la apariencia del puntero, a todos los botones del programa se le ha asignado el cursor llamado "hand2" que es un icono con la forma de una mano, muy conocido de distintos programas y webs.
- Como en todos los anteriores, utiliza el método .grid() para su colocación.

nombredelButton = Button(*parent*, command = *función*, text = *'Texto dentro del botón'*, bg = *'color'*, relief = *'relieve'*, width = *núm*, cursor = '*icono de cursor* ')

nombredelButton.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)

Tabla 10. Estructura de funciones con Button.

```
def botones(num):
    almacenar = Label(miFrame, text = 'Almacenar los datos en:', font = ('Helvetica','13'), bg = 'tomato2')
    almacenar.grid(row = num, column = 0, sticky = 'we', columnspan = 2, padx = 15, pady = 30)
    botonAct = Button(miFrame, text = 'ACTIVOS', command = enviaActivos, cursor = 'hand2')
    botonAct.grid(row = num, column = 2, sticky = 'w', pady = 30)
    botonInac = Button(miFrame, text = 'INACTIVOS', command = enviaInactivos, cursor = 'hand2')
    botonInac.grid(row = num, column = 3, sticky = 'w', pady = 30)
```

Figura 10. Función para crear los botones que enviarán los datos a la tabla correspondiente de la base de datos.

4.2.5. ttk.Combobox

Este formulario cuenta a la hora de modificar los campos de la base de datos con una larga lista de elementos, si se elige sólo uno de ellos sería muy incómodo crear un Radiobutton con cada elemento y ocuparía demasiado espacio; para ello existen las listas desplegables, en la que el operador pulsa el icono con una flecha y se despliegan todas las opciones disponibles, se puede seleccionar una de ellas y al hacerlo dicha lista volverá a plegarse.

El módulo ttk reemplaza a gran parte de la maquinaria original de Tkinter, pero no toda. Debe ser importado y se realiza al principio del programa con el comando "from tkinter import ttk".

La estructura es de esta forma:

- Un Label que solicita al operador que escoja una de las opciones de la lista.
- El ttk.Combobox que cuenta con un argumento no utilizado anteriormente, state, en el cual se especifica el estado de la lista desplegable, si es de sólo lectura o se puede escribir, todos los utilizados en el programa son de sólo lectura, 'readonly'.
- Es necesario además una lista con todos los valores de la lista desplegable, para ello se introducen cada uno de ellos, separados por comas y entre corchetes.
- Como en todos los anteriores, lleva su correspondiente método .grid() para colocarlo en el sitio correspondiente.

nombredelLabel = Label(*parent*, text = *'Elige un elemento:'*, width = *núm*)
nombredelLabel.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)
nombredelCombobox = ttk.Combobox(*parent*, state = *'estado'*, width = *núm*)
nombredelCombobox ['values'] = [*'Valor1','Valor2'*,…,*'ValorN'*]
nombredelCombobox.grid(row = *núm*, column = *núm*, sticky = *'nswe'*, padx = *núm*, pady = *núm*)

Tabla 11. Estructura de funciones con ttk.Combobox.

```
quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
quienmodificar['values'] = nombap
quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)

quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
quemodificar['values'] = ['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
                          'Profesor_Anfitrión_DOE','Email_profesor','Fecha_llegada','Fecha_salida_prevista','Fecha_salida_real',
                          'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación',
                          'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
                          'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
                          'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
```

Figura 11. Ejemplo de ttk.Combobox, éste es utilizado para elegir el campo a modificar de la BBDD.

4.2.6. Menu

La gran mayoría de aplicaciones y programas que utilizamos cuenta con una barra de menú, con elementos como: “Archivo”, “Editar”, “Ver”, etc. Al pulsar en uno de estos elementos aparece una lista en cascada con las opciones dentro de esa pestaña. En este programa también he añadido esa barra en la que está la opción de editar campos, eliminar registros, volver al formulario, mostrar la lista de usuarios a punto de dar de baja y salir del programa.

He utilizado este tipo de menú para las diferentes opciones para la gestión de la base de datos, introducir, modificar y eliminar datos, pero la creación de cada pestaña del menú sigue la misma estructura:

- Lo primero es crear la barra del menú, en este caso se hace directamente en la ventana y no en ningún frame.
- Se configura la ventana para crear un menú con las características de la barra de menú.
- Se crean los botones de la barra, en el ejemplo de estructura (Tabla 12) sólo hay uno, para continuar creando se sigue el mismo procedimiento
- El método .add_cascade() sirve para crear una cascada en la que se pueden incluir distintos elementos, de momento sólo tiene un label con el nombre correspondiente.
- El método .add_command() sirve para añadir elementos dentro de la cascada creada anteriormente, en este caso cuentan con un label con un texto, y un command en el que indicamos la función que va a realizar, dicha función estará definida previamente en el programa.

barraMenu = Menu(*ventana*)
ventana.config(menu = *barraMenu*)
nombreMenu = Menu(*barraMenu*, tearoff = *núm*)
barraMenu.add_cascade(label = *‘Nombre del menú’*, menu = *nombreMenu*)
nombreMenu.add_command(label = *'Nombre primera opción'*, command = *volverFormulario*)

Tabla 12. Estructura de funciones con Menu.

```
barraMenu = Menu(raiz)
raiz.config(menu = barraMenu)
dardebaja = Menu(barraMenu, tearoff = 0)
barraMenu.add_command(label = 'Formulario', command = volverFormulario)
barraMenu.add_command(label = 'Editar BBDD', command = editarBD)
barraMenu.add_command(label = 'Eliminar registros', command = borrar)
barraMenu.add_cascade(label = "Ver", menu = dardebaja)
dardebaja.add_command(label = 'Usuarios a punto de dar de baja...', command = usuariosdardebaja)
barraMenu.add_command(label = 'Salir', command = Salir)
```

Figura 12. Código para crear la barra de menú.

4.3. Elección del usuario

Como se ha mencionado en varias ocasiones, en este programa hay tres tipos de perfil de usuario, cada uno con sus correspondientes servicios y sus diferentes campos de datos.

El programa comienza con un texto en el que se pide elegir el tipo de usuario y tres botones de diferentes colores para cada tipo de perfil, independientes unos de otros. Pulsando aparece el formulario correspondiente a cada tipo, como podemos ver en la Figura 13.

Figura 13.Apariencia de los 3 botones de selección de usuario.

4.3.1. Doctorando

El usuario doctorando cuenta con los siguientes campos:

Información personal:	
Nombre y apellidos	Profesor responsable
Dirección email	Email del profesor
Teléfono	Período estancia
Universidad de origen	Fecha de alta
Ciudad	Ubicación en DOE
País	Información adicional
Usuario de la UPVNET	

Tabla 13. Datos de información personal del usuario doctorando.

Servicios de acceso en la tarjeta UPV:		
Comedor	Despachos asociados	Actualizador
Seminario 2 Informática	Seminario 2 Teleco	Seminario 3-1
Seminario 3-2	Seminario 3-3	Seminario 4

Tabla 14. Servicios de acceso para el usuario doctorando.

Servicios de impresora:
Incluir en el grupo becarios-imprimir para el caso de usuario que no tenga cuenta en el dominio UPVNET.
Activar en Papercut para poder imprimir.
Código de fotocopiadora (en caso de requerirlo se les asigna un código número de 4 cifras para poder usar la fotocopiadora ubicada en el piso donde se ubica el usuario.
Activar usuario en la fotocopiadora del piso número: XXX el que esté ubicado el usuario.

Tabla 15. Servicios de impresora para el usuario doctorando.

Otros recursos:
Si se le proporciona ordenador o lo trae el propio usuario

Tabla 16. Otros recursos para el usuario doctorando.

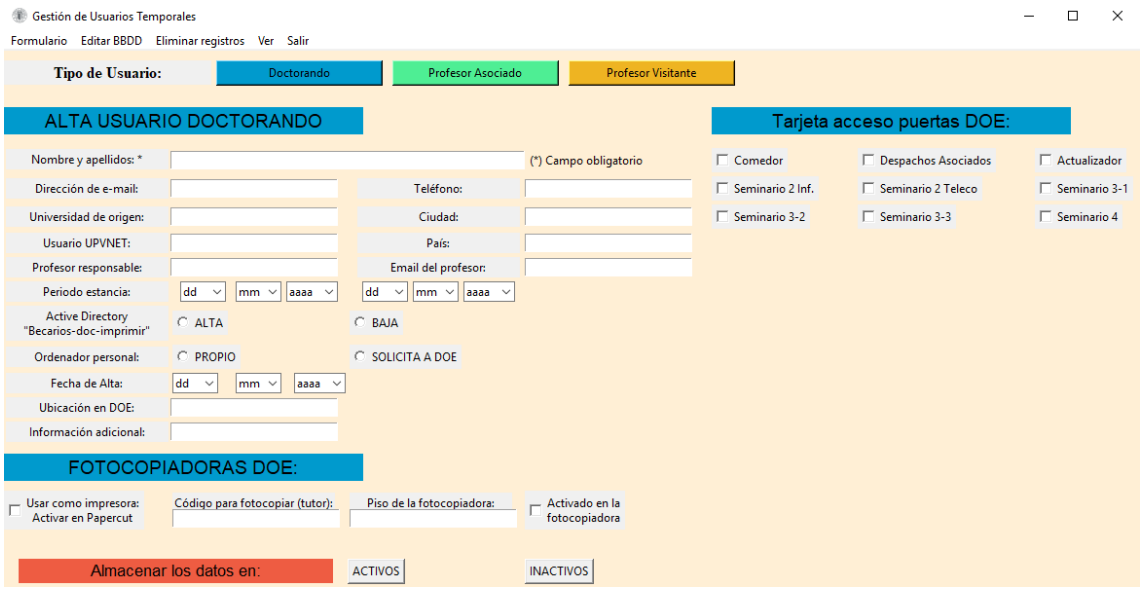

Figura 14. Aspecto de la pantalla para el usuario doctorando.

4.3.2. Profesor asociado

El usuario profesor asociado cuenta con los siguientes campos:

Información personal:	
Nombre y apellidos	Usuario de la UPVNET
Dirección email	Campus UPV en el que realiza la docencia
Teléfono	Fecha de alta
Universidad de origen	Ubicación en el DOE
Ciudad	Información adicional
País	

Tabla 17. Datos de información personal para el usuario profesor asociado.

Servicios de acceso en la tarjeta UPV:		
Comedor	Despachos asociados	Actualizador
Laboratorios	Taquilla	Seminario 2 Informática
Seminario 2 Teleco	Seminario 3-1	Seminario 3-2
Seminario 3-3		

Tabla 18. Servicios de acceso para el usuario profesor asociado.

Servicios de impresora:
Incluir en el grupo becarios-imprimir para el caso de usuario que no tenga cuenta en el dominio UPVNET.
Activar en Papercut para poder imprimir.
Código de fotocopiadora (en caso de requerirlo se les asigna un código número de 4 cifras para poder usar la fotocopiadora ubicada en el piso donde se ubica el usuario.
Activar usuario en la fotocopiadora del piso número: XXX el que esté ubicado el usuario.

Tabla 19. Servicios de impresora para el usuario profesor asociado.

Otros recursos:
Dar de alta en Alfresco.

Tabla 20. Otros recursos para el usuario profesor asociado.

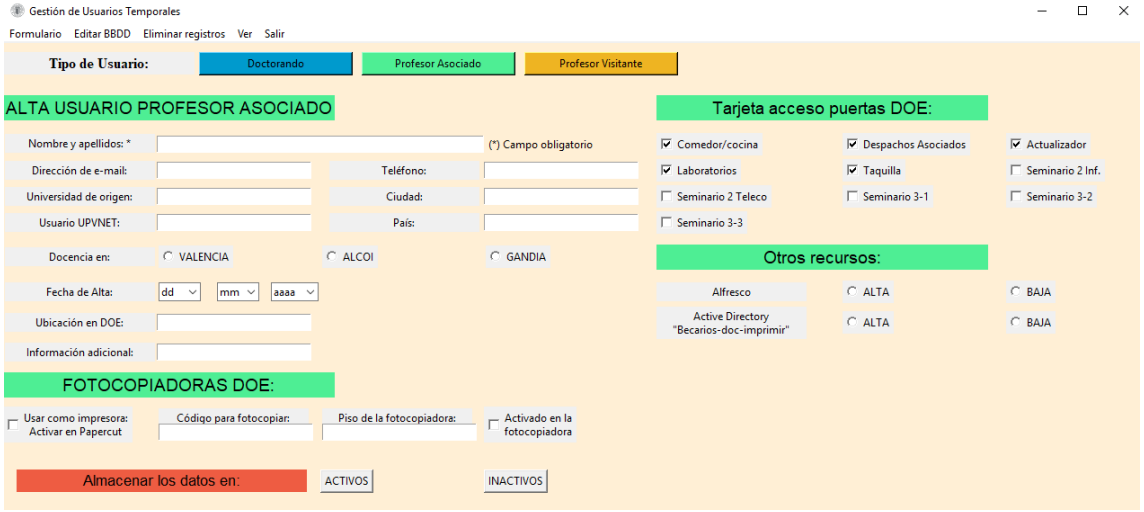

Figura 15. Aspecto de la pantalla del usuario profesor asociado.

4.3.3. Profesor visitante

El usuario profesor visitante cuenta con los siguientes campos:

Información personal:	
Nombre y apellidos	Profesor responsable
Dirección email	Email del profesor
Teléfono	Período estancia
Universidad de origen	Fecha de alta
Ciudad	Ubicación en DOE
País	Información adicional
Usuario de la UPVNET si lo tiene, si no, documento de identificación.	

Tabla 21. Datos de información personal del usuario profesor visitante.

Servicios de acceso en la tarjeta UPV:		
Comedor	Despachos asociados	Actualizador
Seminario 4	Otros	

Tabla 22. Servicios de acceso para el usuario profesor visitante.

Otros recursos:
Si se le proporciona ordenador o lo trae el propio usuario.
Asignar usuario temporal.

Tabla 23. Otros recursos para el usuario profesor visitante.

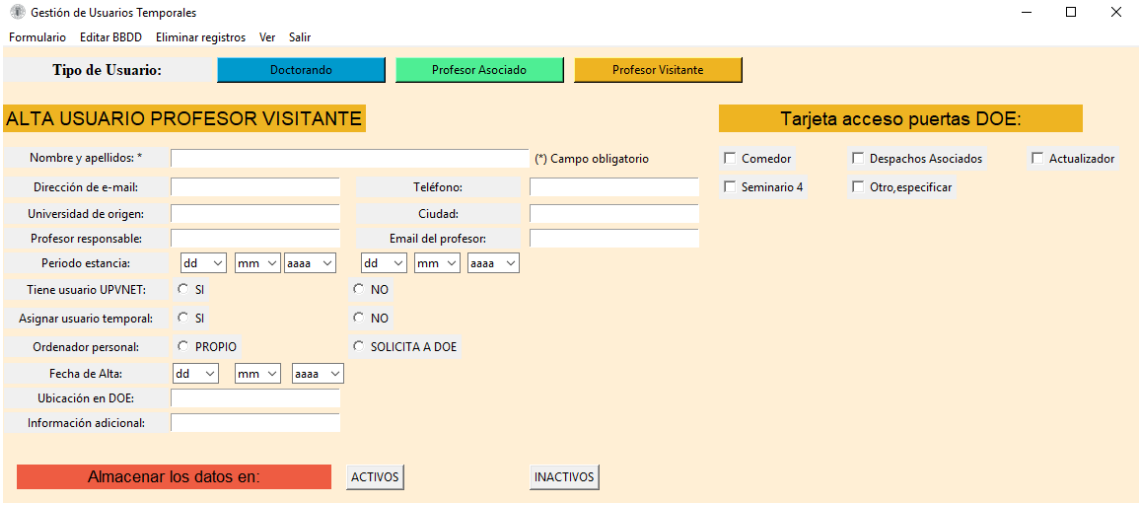

Figura 16. Aspecto de la pantalla del usuario profesor visitante.

4.4. Recogida de los datos

Actualmente en el DOE cuentan con un Excel en el que se almacenan los datos de cada usuario, dicha hoja de cálculo se compone de dos hojas: una llamada “Activos” en la que constan los usuarios visitantes que están actualmente en el DOE; otra hoja titulada “Inactivos” en la que constan los usuarios visitantes que ya no se encuentran en el DOE.

En la base de datos, la cual explicaré detalladamente más adelante, existen dos tablas (igual que en el Excel) una de activos y otra de inactivos. Los tres perfiles de usuario cuentan con dos botones en la parte inferior. La utilidad de estos botones es la de enviar los datos introducidos en el formulario a la tabla correspondiente, ya sea la de activos o la de inactivos.

La obtención de todos los datos introducidos en el formulario se lleva a cabo en una sola función, que voy a explicar a continuación:

Al principio del programa (fuera de las funciones) declaramos todas las variables de control. Las que vayan a contener una cadena de caracteres se declaran como StringVar(), que quiere decir que es una cadena de texto que va a ir variando. Por otro lado, se declaran como IntVar() las variables que van a almacenar números enteros, como es el caso de los Checkbutton y Radiobutton. Las variables en las que se van a introducir fechas, se declaran como datetime (Figura 17).

```
eleccion = ''
n = StringVar()
tlf = StringVar()
uniorigen = StringVar()
ciudad = StringVar()
pais = StringVar()
e = StringVar()
u = StringVar()
prof = StringVar()
emailprof = StringVar()
alta = datetime
fechent = datetime
fechsal = datetime
ubicacion = StringVar()
tienordenador = IntVar()
tieneupvnet = IntVar()
opcionuni = IntVar()
comedor = IntVar()
despachos = IntVar()
actualizar = IntVar()
sem4 = IntVar()
seminf = IntVar()
semtel = IntVar()
sem31 = IntVar()
sem32 = IntVar()
sem33 = IntVar()
laboratorios = IntVar()
taquilla = IntVar()
papercut = IntVar()
fotocopiadora = IntVar()
pisofoto = StringVar()
codfoto = StringVar()
alfresco = IntVar()
directory = IntVar()
otro = IntVar()
otromotivo = StringVar()
infoadicional = StringVar()
usuariotemporal = IntVar()
```

Figura 17. Declaración de las variables de control.

Los Entrys, Checkbuttons y Radiobuttons cuentan con una variable de control en la que se almacena la información, como he explicado en la sección 4.2 “Funciones del formulario”. Pero lo que almacenan dichas variables todavía no son cadenas de tipo String, ni números enteros, puesto que son elementos del tipo StringVar e IntVar, para obtenerlos debemos utilizar el método .get().

4.4.1. Método .get()

Este método devuelve el valor de una variable en forma de String, cuando es StringVar(), o de Int, en un IntVar(). También se utiliza para devolver en forma de String el texto que hay escrito en un Entry, la opción elegida en un ttk.Combobox, y el valor de los Checkbuttons y Radiobuttons.

En la función que recoge todos los datos se utiliza dicho método en todas las variables de control y almacena dicho valor String en una nueva variable que va a ser la que utilizaremos en SQLite para enviarla a nuestra base de datos. Para el caso de los Entry no es necesario realizar nada más.

En el caso de los Checkbutton al utilizar el método .get() se obtiene un número entero, pero a la hora de enviarlo a la base de datos no queremos enviar un número, sino un String que diga “Sí” o “No”, para ello se utiliza un condicional como se ve en la Tabla 24.

```
variableparaenviar = variable.get()
if variableparaenviar == 1:
    variableparaenviar = 'Sí'
else:
    variableparaenviar = 'No'
```

Tabla 24. Obtención del valor de la variable de control en un Checkbutton y su preparación para enviarla en formato String.

Para un Radiobutton el proceso es muy similar, sólo que en este tipo de botón existe una posibilidad, que no se haya pulsado ninguno de los botones y el valor será una cadena vacía, con lo cual también tenemos que tener en cuenta ese posible caso, y esto se realiza mediante condicionales como vemos en la Tabla 25.

```
variableparaenviar = variable.get()
if variableparaenviar == 1:
    variableparaenviar = 'Opción1'
elif variableparaenviar == 2:
    variableparaenviar = 'Opción2'
else:
    variableparaenviar = ' '
```

Tabla 25. Obtención del valor de la variable de control en un Radiobutton y su preparación para enviarla en formato String.

4.4.2. Fechas en el formulario

El módulo datetime proporciona clases para manipular fechas y horas de maneras simples y complejas. Si bien se admite la aritmética de fecha y hora, el objetivo de la implementación es la extracción eficiente de atributos para el formato y la manipulación de los resultados.

Se debe importar este módulo al principio del programa para poder utilizar sus clases.

Las clases utilizadas son:

"%d%m%Y"	Especifica el formato en el que va a estar la fecha, hay muchos tipos, pero en el programa únicamente utilizo éste, %d es la clave para definir el día del mes, %m es el número del mes y %Y es el año en formato largo, con los 4 dígitos.
datetime.strptime(*núm*, *formato*)	Sirve para convertir una cadena a un objeto datetime, dentro del paréntesis se introduce la cadena con los números y el formato que tiene la cadena.
datetime.now()	Crea un objeto datetime con la fecha y hora actual.

Tabla 26. Clases utilizadas del módulo datetime [14].

Una vez conocidas las clases ya es posible entender las funciones con fechas. Para introducir en el formulario una fecha se hace mediante ttk.Combobox (Tabla 11). Se utilizan tres distintos, para día, mes y año. Obtengo el valor de cada uno de ellos con el método .get() como se explica en la subsección 4.4.1 “Método .get()” y se concatenan los 3 datos en una única cadena, la cual será la que introduzcamos en el argumento de nuestro datetime.strptime() para convertirlo en un objeto datetime.

Al ser un objeto datetime, se producirán errores si la fecha introducida no es correcta (Ej. 30-02-2018) para remediarlo y que el programa se ejecute se utilizan excepciones, que capturan el error y envían un mensaje (Figura 18), como se ve en la Tabla 25, así el programa puede continuar a pesar del error.

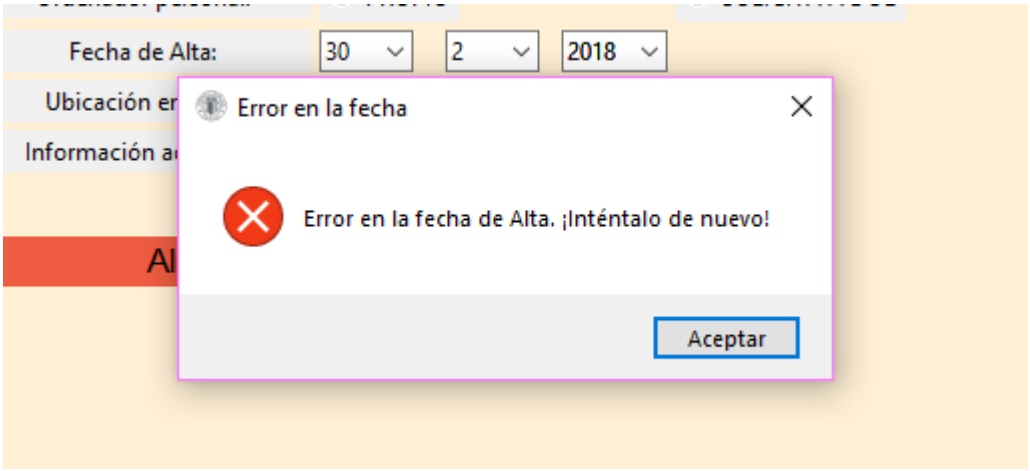

Figura 18. Mensaje de error al introducir una fecha no válida.

Las pestañas desplegables, antes de ser pulsadas, muestran mediante las letras "dd", "mm" y "aaaa" lo que debe ser introducido en cada una (día, mes o año) (Figura 19).

Figura 19. Introducción de fechas mediante ttk.Combobox.

Puede darse el caso de que el operador no cuente con la información de las fechas de un usuario, y quiera que el campo fecha no sea completado, es decir, que esté en blanco en la base de datos. Para ello, mediante un condicional if, se especifica si el valor del ttk.Combobox es el predeterminado, si es así, el valor de la variable que es enviada a la base de datos estará en blanco y no tendrá que ser pasado a objeto datetime.

```
formato = "%d%m%Y"
try:
        if dias.get() == 'dd' or meses.get() == 'mm' or anos.get() == 'aaaa' :
            variable = ' '
        else:
            fecha = dias.get()+meses.get()+anos.get()
            variable = datetime.strptime(fecha, formato)
except:
        messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')
```

Tabla 27. Función para obtener un objeto datetime.

4.5. Creación de la base de datos e introducción de datos

En esta parte del trabajo es en la que empiezo a trabajar con la biblioteca SQLite, voy a comenzar explicando cómo se crea una base de datos en Python con SQLite y a continuación cómo introducir datos en ella.

4.5.1. Crear la base de datos

Lo primero de todo será crear la base de datos, para ello, lo primero que hay que hacer es importar en el programa la biblioteca SQLite, esta acción se realiza escribiendo al principio del programa la instrucción: "import sqlite3".

Una vez importado ya podemos utilizar todas las clases de la biblioteca. Para crear nuestra base de datos hay que crear una conexión, que tendrá su correspondiente título, la conexión debe de ser cerrada una vez terminada la manipulación de la base de datos.

Ya está creada la base de datos, pero todavía no se puede trabajar con ella. Para poder realizar consultas y manejar sus resultados es necesario crear un cursor, éste será el que permita realizar las consultas correspondientes.

Para almacenar los registros es necesario crear una tabla. En este caso serán dos tablas, una para usuarios activos y otra para usuarios inactivos, ambas con los mismos campos.

En la Figura 20 vemos como se han realizado todos estos pasos en el programa y se han creado las tablas correspondientes.

Cuando ya se ha ejecutado por primera vez el programa, la base de datos y sus tablas se crean, si se vuelve a ejecutar dará un error debido a que está pidiendo que vuelva a crearlas, pero ya lo están, por eso dicha acción está dentro de un try, es decir, siempre que se pueda, creará las tablas, pero cuando éstas ya existan, este paso no será realizado.

```
742 miConexion = sqlite3.connect('Usuarios Temporales')
743 cursor = miConexion.cursor()
744 try:
745     cursor.execute('''create table ACTIVOS(
746     Usuarios text,Nombre_y_apellidos text ,Email text,Teléfono integer,Universidad_de_origen text,Ciudad text,
747     País text,Universidad_profesor_asociado text,Profesor_Anfitrión_DOE text,Email_profesor text,
748     Fecha_llegada datetime,Fecha_salida_prevista datetime,Fecha_salida_real datetime,
749     Ubicacion_DOE text,Tipo_ordenador text,Alfresco text,Informacion_adicional text,
750     Usuario_UPVNET_o_Identificación text,Asignar_usuario_temporal text,AD_Imprimir text,
751     Papercut text,Activado_en_fotocopiadora text,Codigo text,Activar_fotocop_piso text,
752     Comedor text,Dpch_Asociados text,Seminario_Informática text,Seminario_Teleco text,
753     Seminario_3_1 text,Seminario_3_2 text,Seminario_3_3 text,Seminario_4 text,
754     Actualizador text,Laboratorios text,Taquillas text,Otros_Indicar text)''')
755
756     cursor.execute('''create table INACTIVOS(
757         Usuarios text,Nombre_y_apellidos text ,Email text,Teléfono integer,Universidad_de_origen text,Ciudad text,
758     País text,Universidad_profesor_asociado text,Profesor_Anfitrión_DOE text,Email_profesor text,
759     Fecha_llegada datetime,Fecha_salida_prevista datetime,Fecha_salida_real datetime,
760     Ubicacion_DOE text,Tipo_ordenador text,Alfresco text,Informacion_adicional text,
761     Usuario_UPVNET_o_Identificación text,Asignar_usuario_temporal text,AD_Imprimir text,
762     Papercut text,Activado_en_fotocopiadora text,Codigo text,Activar_fotocop_piso text,
763     Comedor text,Dpch_Asociados text,Seminario_Informática text,Seminario_Teleco text,
764     Seminario_3_1 text,Seminario_3_2 text,Seminario_3_3 text,Seminario_4 text,
765     Actualizador text,Laboratorios text,Taquillas text,Otros_Indicar text)''')
766
767 except:
768     print()
```

Figura 20. Creación de la conexión, cursor y tablas.

4.5.2. Introducción de datos

Una vez obtenidos los datos del formulario como se muestra en la sección 4.4 "Recogida de los datos" es la hora de introducirlos en su correspondiente campo dentro de la tabla seleccionada.

Dicha acción se realiza con el comando de SQLite "insert into *nombreTabla*", como lo que vamos a introducir en los campos no son directamente los valores, sino que son variables, en *values* tenemos que poner tantos "?" como campos haya en la tabla (en los campos que vayan a ser completados con los datos dentro de una variable) y a continuación se introducen los nombres de todas las variables, en el orden correspondiente en el que deban ir dentro de la tabla. Dependiendo

del botón pulsado en el formulario, se trabajará con la tabla ACTIVOS, Figura 21, o INACTIVOS, Figura 22. Para que los cambios se realicen, los comandos de SQLite deben terminar siempre con *NombreConexion*.commit().

```
cursor.execute('insert into ACTIVOS values(?,?,?,?,?,?,?,?,?,?,?,?,"",?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(
eleccion,nombre,email,telefon,origen,ciud,p,universidad,profesor,correoprofe,alta,fechsal,ubi,tieneordenador,alf,
infadi,upv,usutem,direc,paper,fotocop,cod,piso,comed,desp,semiinf,semitel,semi31,semi32,semi33,semi4,act,lab,taq,otr,))

miConexion.commit()
```

Figura 21. Introducción de los datos en la tabla ACTIVOS.

```
cursor.execute('insert into INACTIVOS values(?,?,?,?,?,?,?,?,?,?,?,?,"",?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(
eleccion,nombre,email,telefon,origen,ciud,p,universidad,profesor,correoprofe,alta,fechsal,ubi,tieneordenador,alf,
infadi,upv,usutem,direc,paper,fotocop,cod,piso,comed,desp,semiinf,semitel,semi31,semi32,semi33,semi4,act,lab,taq,otr,))

miConexion.commit()
```

Figura 22. Introducción de los datos en la tabla INACTIVOS.

4.6. Gestión de la base de datos

Una vez creada la base de datos y la interfaz, introducir en ella los datos es cómodo e intuitivo. Editar y eliminar dichos valores sería muy sencillo utilizando un editor de base de datos cualquiera.

Mi objetivo ha sido evitar utilizar programas externos y poder gestionar la base de datos desde el mismo programa. Para ello, en la barra de menú de la ventana del programa he creado distintos botones (Figura 23), cada uno de ellos lleva a una pantalla diferente. Al principio de cada una de ellas se elimina el frame anterior junto a todos sus widgets y se crea uno nuevo con los nuevos widgets necesarios.

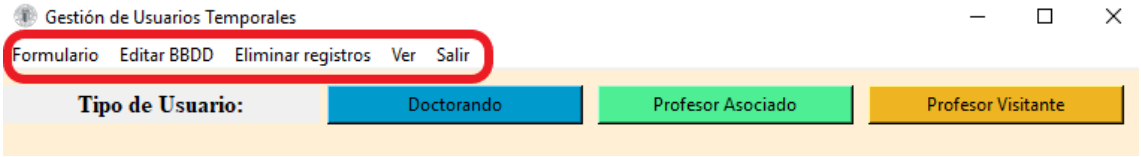

Figura 23. Elementos de la barra de menú.

"Formulario" es la pantalla que aparece por defecto al ejecutar el programa. Contiene todos los elementos explicados anteriormente en la sección 4.3 "Elección de usuario". Este botón sólo hará falta pulsarlo cuando esté en una de las otras pantallas. Sirve para volver a rellenar un formulario e introducir un nuevo usuario a la base de datos.

4.6.1. Editar base de datos

Una aplicación importante es poder editar los datos de la base de datos. Para ello se crea una nueva pantalla, que para distinguirla tendrá el fondo de un color distinto y unos widgets diferentes.

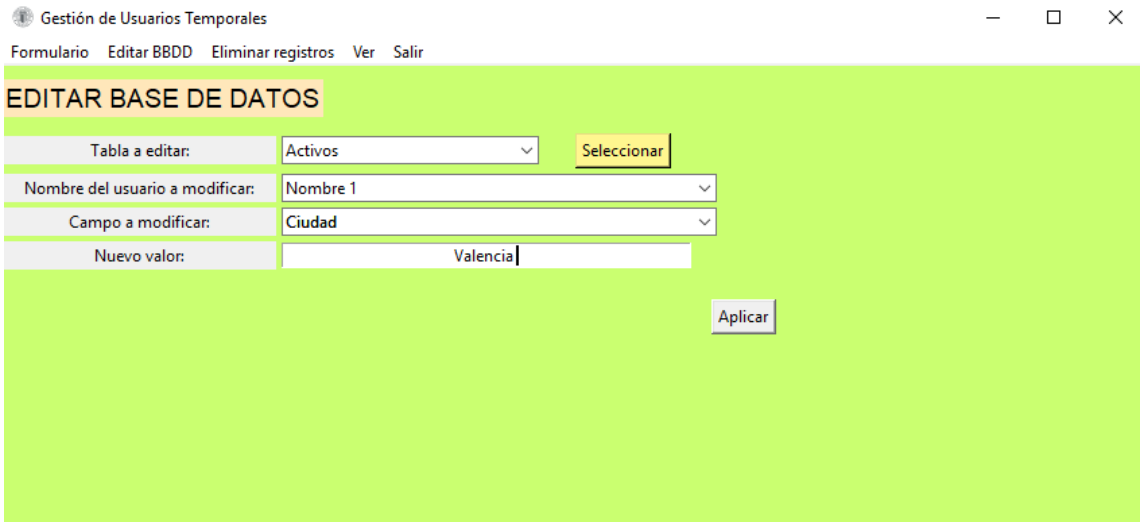

Figura 24. Aspecto de la pantalla Editar base de datos.

Como muestra la Figura 24, comienza seleccionando la tabla que se va a editar; a continuación en la siguiente lista desplegable, aparecerán todos los nombres de los usuarios que se hayen registrados en dicha tabla. Se selecciona el nombre del usuario a editar y posteriormente el campo que se desea modificar, se escribe en el cuadro inferior el nuevo valor para dicho campo y por último se presiona el botón Aplicar para realizar el cambio.

Para explicar la estructura interna de dicha función, lo primero es presentar los comandos de SQLite necesarios para ello (Tablas 28 y 29).

SELECT	**Sirve para recuperar los datos dentro de una tabla**
Recupera los datos del campo *NombreCampo* de la tabla *NombreTabla.*	
Código	cursor.execute('select *NombreCampo* from *NombreTabla*')

Tabla 28. Definición, funcionamiento y código del comando SELECT.

UPDATE	**Actualiza el valor de un dato**
En el registro al que pertenece *Nombre* dentro del campo *Nombre_y_apellidos*, se actualiza el valor del campo *CampoModificado*, por el nuevo valor *NuevoValor*. Los *"?"* son para indicar que es una variable lo que se va a introducir, como se vió en la subsección 4.5.2 "Introducción de datos".	
Código	cursor.execute('update *NombreTabla* set *CampoModificado* = (?) where *Nombre_y_apellidos* = (?)',(*NuevoValor*,*Nombre*,))

Tabla 29. Definición, funcionamiento y código del comando UPDATE.

La función principal elimina frames anteriores y crea uno nuevo para utilizar en dicha pantalla, todos los widgets estarán dentro de este nuevo frame (Figura 25).

```
#-----------------------------------------Editar Base de Datos-----------------------------
def editarBD():
    global miFrame, miFrameEditar,miFrameBorrar,miFrame2
    try:
        miFrameEditar.destroy()
    except:
        print()
    try:
        miFrame.destroy()
        miFrame2.destroy()
    except:
        print()
    try:
        miFrameBorrar.destroy()
    except:
        print()
    miFrameEditar = Frame(raiz)
    miFrameEditar.config(width = 650, height = 350)
    miFrameEditar.config(bg = 'DarkOliveGreen1')
    miFrameEditar.pack(fill = 'both', expand = True)
    raiz.geometry("850x350")
```

Figura 25. Eliminación de Frames existentes y creación de uno nuevo para utilizar en la nueva pantalla.

La función principal cuenta a su vez con subrutinas dentro de ella. Lo primero es seleccionar la tabla que se va a editar y para ello se utiliza un ttk.Combobox. Una vez seleccionada se procede a trabajar con la tabla correspondiente (Figura 26). Voy a explicar el funcionamiento para la tabla ACTIVOS, ya que para la otra simplemente hay que sustituir la palabra ACTIVOS por INACTIVOS en los comandos de SQLite.

```
    def eligeTabla():
        actoinact = tablaaeditar.get()
        if actoinact == 'Activos':
            editarActivos()
        elif actoinact == 'Inactivos':
            editarInactivos()
    labelEditar = Label(miFrameEditar, text = 'EDITAR BASE DE DATOS', font = ('Helvetica','14'),bg = 'wheat1')
    labelEditar.grid(row = 0, column = 0, sticky = 'w', columnspan = 2, pady = 10)
    labeltabla = Label(miFrameEditar, text = 'Tabla a editar:', width = 28)
    labeltabla.grid(row = 1, column = 0, sticky = 'w', padx = 2, pady = 2)

    tablaaeditar = ttk.Combobox(miFrameEditar, state='readonly', width = 28)
    tablaaeditar['values'] = ['Activos','Inactivos']
    tablaaeditar.grid(row = 1, column = 1, sticky = 'w', padx = 2, pady = 2)
    selecboton = Button(miFrameEditar, command = eligeTabla, text = 'Seleccionar', relief = 'raised' , bg = 'khaki1', cursor = 'hand2')
    selecboton.grid(row = 1, column = 2, sticky = 'w', padx = 25, pady = 2)
```

Figura 26. Función encargada de seleccionar la tabla con la que se va a trabajar.

Al elegir la tabla aparecen nuevos widgets:

Dos ttk.Combobox: el primero obtiene sus valores del campo *Nombre_y_apellidos*, que con el comando SELECT, obtiene todos los registros dentro de dicha tabla. El segundo es una lista desplegable con todos los campos de la tabla (Figura 27).

Para asociar un evento con una función utilizo el método .bind(); esto significa que el programa detectará cuándo se ha seleccionado un nuevo elemento en la lista desplegable. Cuando ocurra se ejecutará la función *nuevaseleccion* (Figura 28). Esta función sirve para reconocer si en el campo se va a introducir texto o una fecha, para el primer caso el nuevo valor se introducirá mediante un Entry, mientras que para el otro aparecerán tres listas desplegables para seleccionar: día, mes y año.

```
def editarActivos():
    global nom,nuevo,elec,quemodificar,quienmodificar,insertnuevo
    cursor.execute('select Nombre_y_apellidos from ACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    nombap = []
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        nombap.append(nombres)

    quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
    quienmodificar['values'] = nombap
    quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)
    nombreusuario = Label(miFrameEditar, text = 'Nombre del usuario a modificar:', width = 28)
    nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
    campoamodificar = Label(miFrameEditar, text = 'Campo a modificar:', width = 28)
    campoamodificar.grid(row = 3, column = 0, sticky = 'w', padx = 2, pady = 2)
    quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
    quemodificar['values'] = ['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
                              'Profesor_Anfitrión_DOE','Email_profesor','Fecha_Llegada','Fecha_salida_prevista','Fecha_salida_real',
                              'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación',
                              'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
                              'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
                              'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
    quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
    quemodificar.bind("<<ComboboxSelected>>", nuevaseleccion)
```

Figura 27. Creación de las dos listas desplegables para seleccionar qué y a quién editar.

```
def nuevaseleccion(event):
    global nuevo,diasEditarAct,mesesEditarAct,anosEditarAct,insertnuevo
    labelnuevo = Label(miFrameEditar, text = 'Nuevo valor:', width = 28)
    labelnuevo.grid(row = 4, column = 0, sticky = 'w', padx = 2, pady = 2)
    try:
            insertnuevo.destroy()
    except:
            print()
    try:
            diasEditarAct.destroy()
            mesesEditarAct.destroy()
            anosEditarAct.destroy()
    except:
            print()
    if quemodificar.get() == 'Fecha_Llegada' or quemodificar.get() == 'Fecha_salida_prevista' or quemodificar.get() == 'Fecha_salida_real':

        diasEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
        diasEditarAct['values'] = listadias
        diasEditarAct.grid(row = 4, column = 1, sticky = 'w', padx = 2, pady = 2)
        diasEditarAct.set('dd')
        mesesEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
        mesesEditarAct['values'] = listameses
        mesesEditarAct.grid(row = 4, column = 1, padx = 2, pady = 2)
        mesesEditarAct.set('mm')
        anosEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 5)
        anosEditarAct['values'] = listaanos
        anosEditarAct.grid(row = 4, column = 1, sticky = 'e', padx = 2, pady = 2)
        anosEditarAct.set('aaaa')
    else:
        insertnuevo = Entry(miFrameEditar, textvariable = nuevo,justify = 'center', width = 50)
        insertnuevo.grid(row = 4, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)

    boton = Button(miFrameEditar, command = Aceptar, text = 'Aplicar', cursor = 'hand2')
    boton.grid(row = 5, column = 3, sticky = 'w', padx = 5, pady = 20)
```

Figura 28. Función “nuevaseleccion”.

Una vez completados todos los datos e introducido el nuevo valor que va a tomar el campo a actualizar, queda realizar dicha acción. Para ello al pulsar el botón “Aplicar” se ejecuta la función correspondiente (Figura 29), aquí es donde se obtienen mediante el método .get() los String con el nuevo valor y el campo en el que se va a sustituir. Es ahora cuando por medio de condicionales, se identificará dónde debe ser actualizado el valor y se ejecutará mediante el comando UPDATE.

```
def Aceptar():
    global nuevo,elec,nom,quemodificar,quienmodificar,insertnuevo,diasEditarAct,mesesEditarAct,anosEditarAct
    new = nuevo.get()
    newfechas = diasEditarAct.get()+mesesEditarAct.get()+anosEditarAct.get()
    try:
        insertnuevo.delete(0, last = len(new))
    except:
        print()
    try:
        diasEditarAct.set('dd')
        mesesEditarAct.set('mm')
        anosEditarAct.set('aaaa')
    except:
        print()
    elec = quemodificar.get()
    nom = quienmodificar.get()
    if elec=='Usuarios':
        cursor.execute('update ACTIVOS set Usuarios = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Nombre_y_apellidos':
        cursor.execute('update ACTIVOS set Nombre_y_apellidos = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Email':
        cursor.execute('update ACTIVOS set Email = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Teléfono':
        cursor.execute('update ACTIVOS set Teléfono = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Universidad_de_origen':
        cursor.execute('update ACTIVOS set Universidad_de_origen = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
```

Figura 29. Algunos elementos en la función encargada de actualizar los registros.

4.6.2. Eliminar registros

Al igual que para editar la base de datos, esta pantalla tendrá el fondo de un color diferente, y cuenta con widgets diferentes.

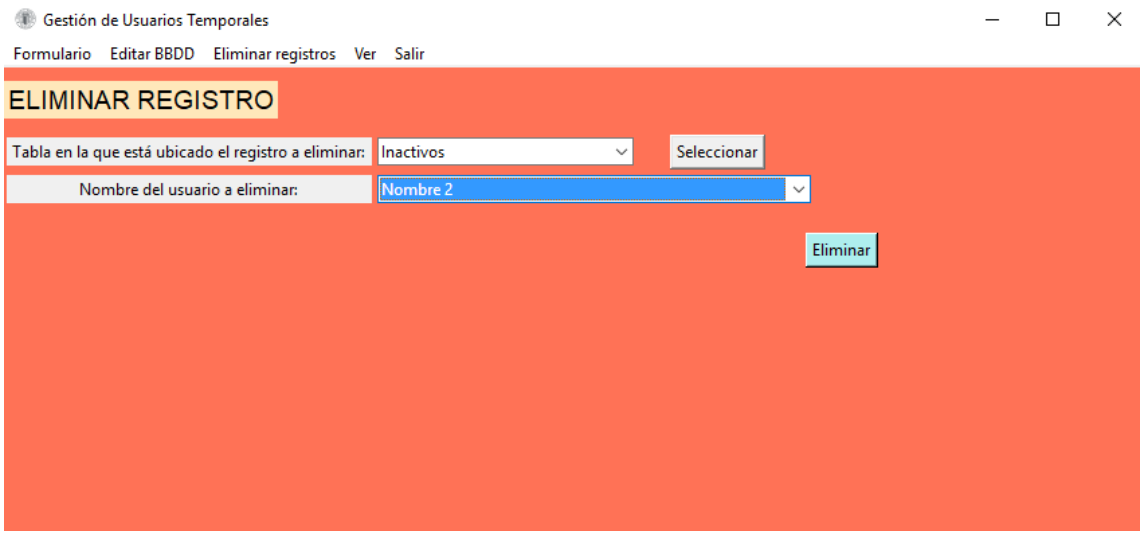

Figura 30. Aspecto de la pantalla Eliminar registros.

Como muestra la Figura 30, se comienza seleccionando la tabla que se va a editar y a continuación en la siguiente lista desplegable aparecerán todos los nombres de los usuarios que haya registrados en dicha tabla. Se selecciona el nombre del usuario a eliminar. Al pulsar el botón “Eliminar” se borrará por completo el registro al que pertenece dicho nombre en la base de datos.

Esta función cuenta con comandos de SQLite, el comando SELECT, explicado en la Tabla 28 y un nuevo comando llamado DELETE (Tabla 30).

DELETE	**Sirve para eliminar un registro**
Elimina el registro al que pertenece *Nombre* dentro del campo *Nombre_y_apellidos* de la tabla *NombreTabla.*	
Código	cursor.execute('delete from *NombreTabla* where *Nombre_y_apellidos* = (?)',(*Nombre*,))

Tabla 30. Definición, funcionamiento y código del comando DELETE.

La función principal elimina frames anteriores y crea uno nuevo para utilizar en dicha pantalla, todos los widgets estarán dentro de este nuevo frame. (Figura 31).

```
def borrar():
    global miFrame, miFrameEditar,miFrameBorrar,miFrame2
    try:
        miFrameBorrar.destroy()
    except:
        print()
    try:
        miFrame.destroy()
        miFrame2.destroy()
    except:
        print()
    try:
        miFrameEditar.destroy()
    except:
        print()
    miFrameBorrar = Frame(raiz)
    miFrameBorrar.pack(fill = 'both', expand = True)
    miFrameBorrar.config(width = 650, height = 350)
    miFrameBorrar.config(bg = 'coral1')

    raiz.geometry("850x350")
```

Figura 31. Eliminación de frames existentes y creación de uno nuevo para utilizar en la nueva pantalla.

La primera subrutina dentro de la función principal sirve para elegir la tabla con la que se va a trabajar, se selecciona mediante un ttk.Combobox (Figura 32).

```
    def eligeTabla():
        actoinact = tablaaeliminar.get()
        if actoinact == 'Activos':
            eliminarActivos()
        elif actoinact == 'Inactivos':
            eliminarInactivos()
    labelEliminar = Label(miFrameBorrar, text = 'ELIMINAR REGISTRO', font = ('Helvetica','14'),bg = 'wheat1')
    labelEliminar.grid(row = 0, column = 0, sticky = 'w', columnspan = 2, pady = 10)
    labeltabla = Label(miFrameBorrar, text = 'Tabla en la que está ubicado el registro a eliminar:', width = 38)
    labeltabla.grid(row = 1, column = 0, sticky = 'w', padx = 2, pady = 2)

    tablaaeliminar = ttk.Combobox(miFrameBorrar, state='readonly', width = 28)
    tablaaeliminar['values'] = ['Activos','Inactivos']
    tablaaeliminar.grid(row = 1, column = 1, sticky = 'w', padx = 2, pady = 2)
    selecboton = Button(miFrameBorrar, command = eligeTabla, text = 'Seleccionar', cursor = 'hand2')
    selecboton.grid(row = 1, column = 2, sticky = 'w', padx = 25, pady = 2)
```

Figura 32. Función encargada de seleccionar la tabla con la que se va a trabajar.

En este caso únicamente hay una lista desplegable con los nombres de los usuarios (Figura 33), se selecciona el nombre del usuario que va a ser borrado y se pulsa el botón “Eliminar”.

```
def eliminarActivos():
    global quieneliminaract
    nombreusuario = Label(miFrameBorrar, text = 'Nombre del usuario a eliminar:', width = 38)
    nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
    cursor.execute('select Nombre_y_apellidos from ACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    nombap = []
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        nombap.append(nombres)
    quieneliminaract  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
    quieneliminaract ['values'] = nombap
    quieneliminaract .grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)

    selecboton = Button(miFrameBorrar, command = EliminarAct, text = 'Eliminar', bg = 'PaleTurquoise2', cursor = 'hand2')
    selecboton.grid(row = 3, column = 3, sticky = 'w', padx = 5, pady = 20)
def eliminarInactivos():
    global quieneliminarinact
    nombreusuario = Label(miFrameBorrar, text = 'Nombre del usuario a eliminar:', width = 38)
    nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
    cursor.execute('select Nombre_y_apellidos from INACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    nombap = []
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        nombap.append(nombres)
    quieneliminarinact  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
    quieneliminarinact ['values'] = nombap
    quieneliminarinact .grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)

    selecboton = Button(miFrameBorrar, command = EliminarInact, text = 'Eliminar', bg = 'PaleTurquoise2', cursor = 'hand2')
    selecboton.grid(row = 3, column = 3, sticky = 'w', padx = 5, pady = 20)
```

Figura 33. Creación de la lista desplegable con los nombres de los usuarios.

Al pulsar dicho botón se ejecutará la función que contiene el comando DELETE, eliminando así el registro de dicha lista (Figura 34).

```
def EliminarAct():
    global quieneliminaract
    nom = quieneliminaract .get()
    cursor.execute('delete from ACTIVOS where Nombre_y_apellidos = (?)',(nom,))
    miConexion.commit()
    cursor.execute('select Nombre_y_apellidos from ACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    nombap = []
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        nombap.append(nombres)
    quieneliminaract  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
    quieneliminaract ['values'] = nombap
    quieneliminaract.grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)
def EliminarInact():
    global quieneliminarinact
    nom = quieneliminarinact.get()
    cursor.execute('delete from INACTIVOS where Nombre_y_apellidos = (?)',(nom,))
    miConexion.commit()
    cursor.execute('select Nombre_y_apellidos from INACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    nombap = []
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        nombap.append(nombres)
    quieneliminarinact  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
    quieneliminarinact ['values'] = nombap
    quieneliminarinact.grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)
```

Figura 34. Función encargada de eliminar los registros.

4.6.3. Ver usuarios a punto de dar de baja.

Una de las utilidades más importantes de este programa es avisar cuando reste menos de 15 días para que algún usuario deba ser dado de baja. Cuando el campo *Fecha_salida_prevista* de alguno de los registros tenga una diferencia menor o igual a 15 días con respecto al día actual, aparecerá un mensaje que avisará de que hay un usuario que debe ser dado de baja (Figura 35).

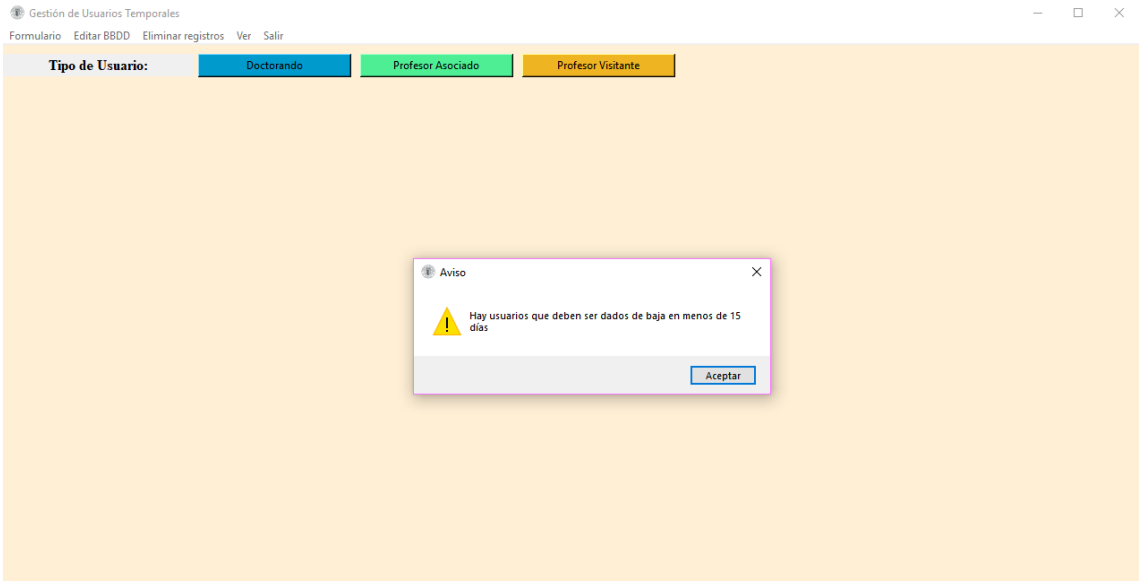

Figura 35. Mensaje de aviso de usuarios a punto de dar de baja.

Para que el programa pueda realizar dicha acción se vuelve a hacer uso del comando SELECT (Tabla 26), almacenándose todos los valores dentro de una tupla, ésta contiene una lista con todos los datos. Para esto se utiliza el método .fetchall(), se efectúa la misma operación para tener todos los valores del campo *Nombre_y_apellidos* en una variable (Figura 36).

```
cursor.execute('select Fecha_salida_prevista from ACTIVOS')
v = cursor.fetchall()
miConexion.commit()
cursor.execute('select Fecha_salida_prevista from INACTIVOS')
x = cursor.fetchall()
miConexion.commit()
cursor.execute('select Nombre_y_apellidos from ACTIVOS')
nombreyapellidos = cursor.fetchall()
miConexion.commit()
cursor.execute('select Nombre_y_apellidos from INACTIVOS')
nombreyapellidos2 = cursor.fetchall()
miConexion.commit()
global nomusuarios, nomusuarios2
fechaactual = datetime.now()
longact = range(len(v))
contador = 0
nomusuarios = []
nomusuarios2 = []
```

Figura 36. Obtención de los valores mediante el comando SELECT.

Una vez obtenidas ambas tuplas se necesita cada valor de forma independiente, es decir, cada fecha y cada nombre del usuario correspondiente a dicha fecha. Se trabaja con un bucle (Figura 37) para recorrer todos los índices de la lista e identificar el campo que buscamos. El uso de bucles en programación es algo muy utilizado en el Grado, por este motivo estoy muy familiarizado con ellos. Se utilizan en asignaturas como Programación, Tratamiento digital de señal, etc.

```
for i in longact:
    st = v[i][0]
    if st != '':
        pas = st.split('-')
        year = pas[0]
        mes = pas[1]
        day = pas[2]
        daycut = day.split(' ')
        dia = daycut[0]
        fechaacomparar = str(dia+mes+year)
        formato = "%d%m%Y"
        fec = datetime.strptime(fechaacomparar, formato)
        diferencia = fec-fechaactual
        if diferencia.days <= 15 and fec !='':
            contador = contador+1
            nya = nombreyapellidos[i][0]
            nomusuarios.append(nya)
if contador >= 1:
    messagebox.showwarning('Aviso','Hay usuarios que deben ser dados de baja en menos de 15 días de la tabla ACTIVOS')
```

Figura 37. Cálculo de la diferencia de días entre la fecha de salida prevista y el día actual.

En la barra de menú hay una pestaña titulada "Ver" que al poner el cursor sobre ella se abre una cascada en la que aparece "Usuarios a punto de dar de baja…" (Figura 38), al pinchar se ejecutará una función que creará una nueva ventana y aparecerá en ella el nombre de cada uno de los usuarios que deben ser dados de baja.

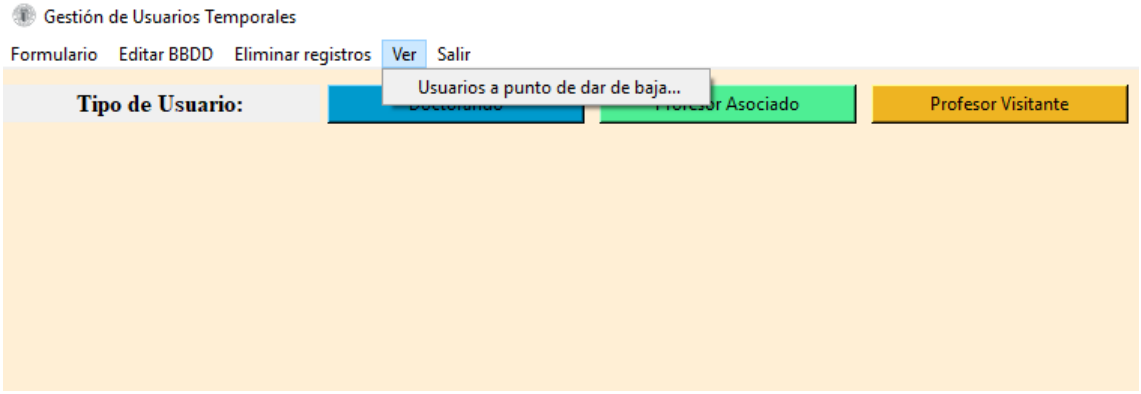

Figura 38. Barra de menú con la opción para ver los usuarios a punto de dar de baja.

Gracias a la función que identifica el nombre de los usuarios en cuestión (Figura 36) ya se conoce el nombre. Ahora mediante Labels habrá que mostrar por pantalla cada uno de ellos (Figura 39).

Figura 39. Ventana con los usuarios a dar de baja.

4.6.4. Salir

Como cualquier aplicación o programa necesita poder cerrarse y salir de éste, para ello se crea esta sencilla función que consiste en una ventana emergente que pregunta si deseamos salir del programa (Figura 40). Si se pulsa "No" simplemente desaparece dicha ventana y si se hace click en "Sí" se cierra la conexión de la base de datos y se elimina la ventana con el método .destroy(). (Figura 41)

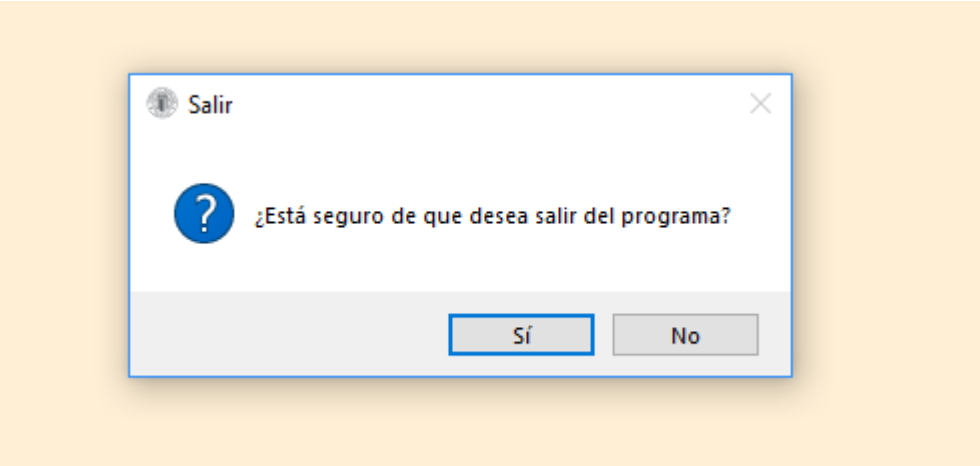

Figura 40. Ventana emergente que pregunta si deseamos salir del programa.

```
def Salir():
    valor = messagebox.askquestion('Salir','¿Está seguro de que desea salir del programa?')
    if valor == 'yes':
        miConexion.close()
        raiz.destroy()
```

Figura 41. Código para la función Salir del programa.

5. Presupuesto

Descripción:	Horas	Precio/hora	Total
Diseño y gestión de una base de datos de usuarios temporales en el Departamento de Organización de Empresas-UPV.	270	20 €	5.400 €

Tabla 31. Descripción del presupuesto por horas de trabajo y esfuerzo.

Importe Neto	*5.400,00 €*
Total I.V.A. 21%	*1134,00 €*
Presupuesto total	***6.534,00 €***

Tabla 32. Presupuesto total del proyecto.

6. Conclusiones.

Cuando me planteé realizar este trabajo, me encontré con la primera dificultad que fue aprender a manejar el lenguaje Python.Para lo cual tuve que dedicar gran esfuerzo y horas en dicho aprendizaje. Encontré material suficiente en Internet para realizar esta tarea, y fueron decenas de vídeos y tutoriales los que tuve que ver para familiarizarme con dicho lenguaje, con la debida toma de datos de los mismos.

Una vez aprendí a utilizar el lenguaje Python, decidí que para que el programa fuera visualmente atractivo, intuitivo y fácil de manejar, quise trabajar con una librería con la que implementar una interfaz gráfica en Python. Esta tarea también me llevó un gran esfuerzo, teniendo en cuenta que tuve que buscar la librería más adecuada y aprender a utilizarla a un nivel avanzado. Para esto recurrí de nuevo a ver una gran cantidad de videos tutoriales y buscar información en internet, además de consultar el manual de usuario oficial de dicha librería.

Posteriormente creada una primera interfaz, tarea que me llevó la mayor parte del tiempo, fue el momento de estudiar el funcionamiento de una base de datos y por supuesto cómo implementarla en Python, lo que me obligó a volver a tener que recurrir a decenas de tutoriales; todo ello me originó la dedicación de muchas horas.

En este punto, el siguiente paso a seguir fue la implementación de la base de datos, y las funciones para introducción, modificación y eliminación de datos. Esta tarea fue complicada porque tuve que adaptar la interfaz a las nuevas funciones y siempre buscando que fuera lo más intuitivo posible para el operador.

Con todo esto realizado era la hora de comprobar el perfecto funcionamiento del programa y detectar posibles errores que pudieran surgir. Después de horas de repaso y comprobaciones, tenía que solucionar los posibles errores que hubieran surgido lo cual me supuso un mayor número de horas de trabajo.

Con el funcionamiento del programa ya resuelto, lo siguiente fue darle a todo el programa un aspecto atractivo, ordenado y de fácil utilización. Por ejemplo el cambio de color en las distintas pantallas, el aplicarle a todos los elementos un mismo tamaño para verse de una forma más estructurada, entre otros.

Para finalizar este trabajo tuve que realizar la memoria, plasmando en ella todo el proceso llevado a cabo en la realización de este proyecto, que ésta fuera lo más comprensible posible y de utilidad para los potenciales usuarios. La finalidad de este programa es la de integrarlo en la página web actual del departamento para que quede operativo. Será el informático de éste departamento el encargado de realizar dicha tarea.

6.1. Ventajas del software utilizado

Para este trabajo hemos decidido utilizar lenguaje Python ya que es uno de los lenguajes actuales con más visión de futuro, en vez del conocido Java, que es el lenguaje que he aprendido en el Grado. A pesar de que ahora existe un entorno de desarrollo libre para Java, no toda plataforma Java es libre. Sun sigue distribuyendo una plataforma Java ejecutable que no es libre, también otras compañías lo hacen y aunque en sí mismo sea libre, puede estar limitado por alguna dependencia de software que no es libre. [17]

6.1.1. Python

- Facilidad de uso. Una persona puede empezar a hacer programas sencillos en Python en muy poco tiempo. A esto contribuye la gestión automática de memoria o las operaciones sencillas de lectura y escritura, a diferencia de otros lenguajes como C.
- Legibilidad del código. La estructura del código es bastante natural y promueve una forma de escribir que facilita su lectura. Esta es una ventaja importante frente a lenguajes dirigidos al mismo sector, como Perl.

- Facilidad de uso en dispositivos. Algunas plataformas como Raspberry Pi están basadas en Python.
- Facilidad de escritura de código asíncrono. Los lenguajes diseñados antes de que las plataformas multiprocesador (o multinúcleo) estuvieran tan generalizadas suelen tener estructuras bastante complicadas para mantener distintos hilos de ejecución; en Python el código asíncrono es bastante sencillo de gestionar.
- Abundancia de bibliotecas. Hay muchas bibliotecas disponibles para extender la funcionalidad básica de Python a cualquier campo.
- Gran base de usuarios. Esto hace que exista mucho código disponible en internet y que los foros de usuarios sean bastante activos, por lo que es fácil encontrar ayuda cuando se necesita [15].

6.1.2. SQLite3

- No requiere configuración
- No se requiere uso de servidor (proceso activo para atender la peticiones)
- Fácilmente portable (multiplataforma windows, linux, mac, dispositivos mobiles, tablets, etc.)
- En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.
- Acceso mucho más rápido.
- Prácticamente cualquier lenguaje y SO lo soportan.
- Registros de Longitud Variable
- Único archivo de Base de datos
- KumbiaPHP soporta SQLite
- Seguridad de los datos (más de 2/3 del código esta dedicado puramente a la prueba y verificación)
- Transaccional (ACID).
- No precisamente es una ventaja pero, los permisos a la BD se realizan por el fichero (no existen usuarios).
- Cuando se consulta se bloquea el fichero pero como todo sucede en milisegundos [19].

6.2. Desventajas del software utilizado

6.2.1. Python

- Python es un lenguaje de programación de tipado dinámico y de la familia de los lenguajes interpretados (conocidos como scripting), esto muchas veces significa que sacrifica rendimiento en favor de flexibilidad y de una simple sintaxis. Por lo cual Python no es la mejor opción para tareas que requieren un rendimiento intensivo, para eso está C++, Go, o algún lenguaje compilado.
- A falta de un compilador, uno debe buscar herramientas de análisis estático para lenguajes de tipos dinámicos, cuya efectividad es severamente restringida por la cantidad de características implícitas y los muchos supuestos que debe hacerse en la ausencia de anotaciones de semántica operacional, e implementar más pruebas para satisfacer a QA. Mientras tanto, un compilador para un lenguaje de tipos estáticos fuertes fácilmente detecta un error en la programación (que de otro modo podría aparecer después) en asignar un valor numérico a una variable de tipo carácter [16].

6.2.2.SQLite3

- Limitaciones en Where: esta limitación está dada por el soporte para clausuras anidadas.
- Falta de Clave Foránea: se hace caso omiso de las claves foráneas; esto quiere decir, cuando se realice la creación de la tabla desde el modo consola, está permitiendo el uso de la clausura, aunque no realizara el chequeo de la misma.
- Falta de documentación en español: si bien ya contamos con una comunidad latino americana de SQLite, sería importante encontrar mucha más documentación, libros, review, etc. como muchos otros motores de bases de datos cuentan hoy en día [18].

6.3. Limitaciones y futuras líneas de trabajo.

Este programa cumple perfectamente con las especificaciones indicadas en el guión del proyecto. Ahora bien, son múltiples las opciones que existen a la hora de crear diferentes interfaces, funciones, etc. Pero todo trabajo tiene un límite de tiempo y hay que tomar decisiones y elegir la más adecuada. Por lo que considero que este trabajo se ajusta a lo requerido por el por el departamento.

Con el uso rutinaro de un programa, es cuando podemos ver que modificaciones se podrían realizar al mismo, de acuerdo a las necesidades y avances que se vayan realizando en el departamento, Como futuras líneas de trabajo, sugiriría que los usuarios cuya fecha de salida sea cumplida pasasen automáticamente a la tabla INACTIVOS y que las notificaciones de que hay usuarios a punto de ser dados de baja fuesen enviadas por email a la persona encargada de gestionar la base de datos.

6.4. Valoración personal

Califico de muy positivo el apredizaje realizado con este trabajo. He sido capaz de aprender a utilizar un nuevo lenguaje de programación a un nivel avanzado de manera autodidacta, gracias a la buena base que poseía de las asignaturas del Grado de programación Java. Por primera vez he trabajado con una base de datos y he comprendido como funcionan y como crearlas y gestionarlas con SQLite.

Me parece que este trabajo me va a ser de utilidad ya que el aprender a programar y crear formularios y manejar interfaces gráficas es algo muy utilizado hoy en día en cualquier sitio y las empresas buscan Graduados en Ingeniería de Tecnologías y Servicios de Telecomunicación para desarrollar este tipo de aplicaciones. Ahora podré incluir en mi CV que he trabajado en el diseño y la gestión de una base de datos, además de conocimiento de un nuevo lenguaje de programación y de un gestor de base de datos.

Estoy contento de que mi trabajo tenga una utilidad real y vaya a servir para que los trabajadores del DOE puedan registrar a los nuevos usuarios y modificar sus datos de una manera sencilla y cómoda.

Elegí este trabajo porque quería que me sirviese de verdad para aprender algo que vaya a poder aplicar a mi vida, y no algo totalmente teórico, y así ha sido. El programa es 100% original y realizado por mi cada línea de código.

Bibliografía

[1] TFG Gestión usuarios temporales DOE.
TFG Gestión usuarios temporales DOE.doc

[2] Departamento de organización de empresas UPV. (19 de agosto 2018).
https://www.doe.upv.es/index.php/departamento/

[3] Python. What is Python? Executive Summary. (5 de julio de 2018).
https://www.python.org/doc/essays/blurb/

[4] Eclipse. Eclipse documentation - Previous Release. What is Eclipse?. (5 de julio de 2018)
https://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm

[5] Devcode. ¿Qué es y por qué aprender SQL?. (5 de julio de 2018).
https://devcode.la/blog/que-es-sql/

[6] SQLite. About SQLite. (5 de julio de 2018).
https://www.sqlite.org/about.html

[7] DOE UPV. Alta usuario Doctorando. (1 de julio de 2018).
https://www.doe.upv.es/?wpfb_dl=22

[8] DOE UPV. Alta usuario Asociado. (1 de julio de 2018).
https://www.doe.upv.es/?wpfb_dl=3

[9] DOE UPV. Alta usuario Visitante. (1 de julio de 2018).
https://www.doe.upv.es/?wpfb_dl=23

[10] Curso de Python. [codigofacilito]. (26 de junio. 2018). [Lista de reproducción]. Recuperado de:
https://www.youtube.com/playlist?list=PLE549A038CF82905F

[11] Curso Python desde 0. [pildorasinformaticas]. (3 de julio. 2018). [Lista de reproducción]. Recuperado de:
https://www.youtube.com/playlist?list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS

[12] Tkinter 8.5 reference: a GUI for Python.
http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html

[13] Python 3 para impacientes. Tkinter: interfaces gráficas en Python. (3 de julio de 2018).
https://python-para-impacientes.blogspot.com/2015/12/tkinter-interfaces-graficas-en-python-i.html

[14] Python. 8.1. datetime — Basic date and time types. (20 de julio de 2018).
https://docs.python.org/3.3/library/datetime.html

[15] Quora. ¿Qué ventajas nos ofrece Python respecto a otros lenguajes de programación?. (5 de julio de 2018).
https://es.quora.com/Qu%C3%A9-ventajas-nos-ofrece-Python-respecto-a-otros-lenguajes-de-programaci%C3%B3n

[16] Quora. ¿Cuáles son las principales debilidades de Python como lenguaje de programación?. (5 de julio de 2018).
https://es.quora.com/Cu%C3%A1les-son-las-principales-debilidades-de-Python-como-lenguaje-de-programaci%C3%B3n

[17] El sistema operativo GNU. Libre pero encadenado. La trampa de Java. (18 de agosto de 2018). https://www.gnu.org/philosophy/java-trap.es.html

[18] Republica. Empresa y economía. SQLite, el motor de base de datos ágil y robusto. (20 de agosto de 2018). http://empresayeconomia.republica.com/aplicaciones-para-empresas/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html

[19] Soy Programador. Conociendo un poco más a SQLite. (20 de agosto de 2018). http://soyprogramador.liz.mx/conociendo-un-poco-ms-a-sqlite/

Anexo

Código del programa

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from datetime import datetime
import sqlite3
import math

raiz = Tk()
raiz.title('Gestión de Usuarios Temporales')
raiz.iconbitmap('logoupv.ico')
raiz.config(bg='papaya whip', width = 650, height = 350)

global miFrame, miFrameEditar,miFrameBorrar,miFrame2
global
eleccion,cuadroNombre,cuadroTlf,cuadroEmail,cuadroorigen,cuadroAlta,fecha_desde,fecha_hasta,cuadrociudad,cuadropais,cuadro
ubi,cuadroUPVnet,cuadroprofres,cuadroemailprof,fecha_desde,fecha_hasta,cuadroAlta,texto,cuadroidentificacion,codigofotocopiar,
pisofotocopia,especificar
global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,laboratorios,taquilla,alfresco,papercut,fotocopiadora,directo
ry,tienordenador,tieneupvnet,usuariotemporal,opcionuni
#-------------------------Declaración de las variables de control--------------------------
eleccion = ''
n = StringVar()
tlf = StringVar()
uniorigen = StringVar()
ciudad = StringVar()
pais = StringVar()
e = StringVar()
u = StringVar()
prof = StringVar()
emailprof = StringVar()
alta = datetime
fechent = datetime
fechsal = datetime
ubicacion = StringVar()
tienordenador = IntVar()
tieneupvnet = IntVar()
opcionuni = IntVar()
comedor = IntVar()
despachos = IntVar()
actualizar = IntVar()
sem4 = IntVar()
seminf = IntVar()
semtel = IntVar()
sem31 = IntVar()
sem32 = IntVar()
sem33 = IntVar()
laboratorios = IntVar()
taquilla = IntVar()
papercut = IntVar()
fotocopiadora = IntVar()
pisofoto = StringVar()
codfoto = StringVar()
alfresco = IntVar()
directory = IntVar()
otro = IntVar()
otromotivo = StringVar()
infoadicional = StringVar()
usuariotemporal = IntVar()
nuevo = StringVar()
elec = StringVar()
name = StringVar()
nomusuarios = []
nomusuarios2 = []
listadias = list(range(1,32))
listameses = list(range(1,13))
listaanos = list(range(2017,2030))
```

```
#----------------Usuario Doctorando----------------
def doctorando():
    global miFrame,miFrame2
    global
eleccion,cuadroNombre,cuadroTlf,cuadroEmail,cuadroorigen,cuadroAlta,fecha_desde,fecha_hasta,cuadrociudad,cuadropais,cuadro
ubi,cuadroUPVnet,cuadroprofres,cuadroemailprof,fecha_desde,fecha_hasta,cuadroAlta,texto,cuadroidentificacion,codigofotocopiar,
pisofotocopia,especificar
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,laboratorios,taquilla,alfresco,papercut,fotocopiadora,directo
ry,tienordenador,tieneupvnet,usuariotemporal,opcionuni
    if eleccion == 'Profesor visitante' or eleccion == 'Profesor asociado':
        eleccion = 'Doctorando'
    try:
        reiniciarEntrys()
    except:
        print()
    try:
        miFrame.destroy()
    except:
        print()
    miFrame = Frame(raiz)
    miFrame.pack(fill = 'both', expand = True)
    miFrame.config(width = 650, height = 350)
    miFrame.config(bg = 'papaya whip')
    labelvisitante = Label(miFrame, text = 'ALTA USUARIO DOCTORANDO', font = ('Helvetica','14'),bg = 'DeepSkyBlue3', width
= 33)
    labelvisitante.grid(row = 1, column = 0, sticky = 'w', columnspan = 3, pady = 12)
    eleccion = 'Doctorando'
    nomape(2)
    telEfono(3)
    email(3)
    uniOrigen(4)
    ciuDad(4)
    paIs(5)
    upvnet(5)
    profeResp(6)
    emailProf(6)
    periodoestancia(7)
    activeDirectory(8,0,2,2)
    ordenadorPersonal(9)
    fechAlta(10)
    ubi(11)
    infoAdicional(12)
    botones(15)
#---------------------------Servicios de acceso usuario Doctorando-----------------------------
    labelTarjetaacceso = Label(miFrame, text = 'Tarjeta acceso puertas DOE:', font = ('Helvetica','14'),bg = 'DeepSkyBlue3', width =
33)
    labelTarjetaacceso.grid(row = 1, column = 4, sticky = 'w', columnspan = 3, padx = 20, pady = 10)
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,papercut,fotocopiadora,pisofoto,codfoto,codigofotocopiar,p
isofotocopia
    Checkbutton(miFrame, text = 'Comedor', variable = comedor,onvalue = 1,offvalue = 0).grid(row = 2, column = 4, sticky = 'w', padx
= 20, pady = 2)
    Checkbutton(miFrame, text = 'Despachos Asociados', variable = despachos,onvalue = 1,offvalue = 0).grid(row = 2, column = 5,
sticky = 'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Actualizador', variable = actualizar,onvalue = 1,offvalue = 0).grid(row = 2, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 2 Inf.', variable = seminf,onvalue = 1,offvalue = 0).grid(row = 3, column = 4, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 2 Teleco', variable = semtel,onvalue = 1,offvalue = 0).grid(row = 3, column = 5, sticky =
'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-1', variable = sem31,onvalue = 1,offvalue = 0).grid(row = 3, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-2', variable = sem32,onvalue = 1,offvalue = 0).grid(row = 4, column = 4, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-3', variable = sem33,onvalue = 1,offvalue = 0).grid(row = 4, column = 5, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 4', variable = sem4,onvalue = 1,offvalue = 0).grid(row = 4, column = 6, sticky = 'w', padx
= 20, pady = 2)
#---------------------------Servicios de fotocopiadora usuario Doctorando--------------------------
    labelfotocopiadoras = Label(miFrame, text = 'FOTOCOPIADORAS DOE:', font = ('Helvetica','14'),bg = 'DeepSkyBlue3', width =
33)
    labelfotocopiadoras.grid(row = 13, column = 0, sticky = 'w', columnspan = 3, pady = 10)
```

```
    Checkbutton(miFrame, text = 'Usar como impresora:\nActivar en Papercut', variable = papercut,onvalue = 1,offvalue = 0).grid(row
= 14, column = 0, sticky = 'w')
    Checkbutton(miFrame, text = 'Activado en la\nfotocopiadora', variable = fotocopiadora,onvalue = 1,offvalue = 0).grid(row = 14,
column = 3, sticky = 'w')
    labelcodigofotocopiar = Label(miFrame, text = 'Código para fotocopiar (tutor):', width = 23)
    labelcodigofotocopiar.grid(row = 14, column = 1, sticky = 'nw', padx = (2,5), pady = 2)
    codigofotocopiar= Entry(miFrame, textvariable = codfoto,justify = 'center', width = 28)
    codigofotocopiar.grid(row = 14, column = 1, sticky = 'sw', padx = (2,5), pady = 2)

    labelpisofotocopia = Label(miFrame, text = 'Piso de la fotocopiadora:', width = 23)
    labelpisofotocopia.grid(row = 14, column = 2, sticky = 'nw', padx = 2, pady = 2)
    pisofotocopia= Entry(miFrame, textvariable = pisofoto,justify = 'center', width = 28)
    pisofotocopia.grid(row = 14, column = 2, sticky = 'sw', padx = 2, pady = 2)
#----------------Usuario Profesor asociado----------------
def asociado():
    global miFrame,miFrame2
    global
eleccion,cuadroNombre,cuadroTlf,cuadroEmail,cuadroorigen,cuadroAlta,fecha_desde,fecha_hasta,cuadrociudad,cuadropais,cuadro
ubi,cuadroUPVnet,cuadroprofres,cuadroemailprof,fecha_desde,fecha_hasta,cuadroAlta,texto,cuadroidentificacion,codigofotocopiar,
pisofotocopia,especificar
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,laboratorios,taquilla,alfresco,papercut,fotocopiadora,directo
ry,tienordenador,tieneupvnet,usuariotemporal,opcionuni,fechent,fechsal
    if eleccion == 'Profesor visitante' or eleccion == 'Doctorando':
        eleccion = 'Profesor asociado'
    try:
        reiniciarEntrys()
    except:
        print()
    try:
        miFrame.destroy()
    except:
        print()
    miFrame = Frame(raiz)
    miFrame.pack(fill = 'both', expand = True)
    miFrame.config(width = 650, height = 350)
    miFrame.config(bg = 'papaya whip')
    labelvisitante = Label(miFrame, text = 'ALTA USUARIO PROFESOR ASOCIADO', font = ('Helvetica','14'),bg = 'SeaGreen2',
width = 33)
    labelvisitante.grid(row = 1, column = 0, sticky = 'w', columnspan = 3, pady = 12)
    eleccion = 'Profesor asociado'
    nomape(2)
    telEfono(3)
    email(3)
    uniOrigen(4)
    ciuDad(4)
    paIs(5)
    upvnet(5)
    eligeuni(6)
    fechAlta(7)
    ubi(8)
    infoAdicional(9)
    botones(12)
#---------------------------Servicios de acceso usuario Profesor asociado-----------------------------
    labelTarjetaacceso = Label(miFrame, text = 'Tarjeta acceso puertas DOE:', font = ('Helvetica','14'),bg = 'SeaGreen2', width = 33)
    labelTarjetaacceso.grid(row = 1, column = 4, sticky = 'w', columnspan = 3, padx = 20, pady = 10)
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,laboratorios,taquilla,alfresco,papercut,fotocopiadora,codfoto,pisof
oto,pisofotocopia,codigofotocopiar
    comedor.set(1)
    despachos.set(1)
    actualizar.set(1)
    laboratorios.set(1)
    taquilla.set(1)
    Checkbutton(miFrame, text = 'Comedor/cocina', variable = comedor,onvalue = 1,offvalue = 0).grid(row = 2, column = 4, sticky =
'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Despachos Asociados', variable = despachos,onvalue = 1,offvalue = 0).grid(row =2, column = 5,
sticky = 'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Actualizador', variable = actualizar,onvalue = 1,offvalue = 0).grid(row = 2, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Laboratorios', variable = laboratorios,onvalue = 1,offvalue = 0).grid(row = 3, column = 4, sticky =
'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Taquilla', variable = taquilla,onvalue = 1,offvalue = 0).grid(row = 3, column = 5, sticky = 'w', padx
= 20, pady = 2)
```

```
    Checkbutton(miFrame, text = 'Seminario 2 Inf.', variable = seminf,onvalue = 1,offvalue = 0).grid(row = 3, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 2 Teleco', variable = semtel,onvalue = 1,offvalue = 0).grid(row = 4, column = 4, sticky =
'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-1', variable = sem31,onvalue = 1,offvalue = 0).grid(row = 4, column = 5, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-2', variable = sem32,onvalue = 1,offvalue = 0).grid(row = 4, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 3-3', variable = sem33,onvalue = 1,offvalue = 0).grid(row = 5, column = 4, sticky = 'w',
padx = 20, pady = 2)
#---------------------------Otros recursos usuario Profesor asociado-----------------------------
    labelOtrosrecursos = Label(miFrame, text = 'Otros recursos:', font = ('Helvetica','14'),bg = 'SeaGreen2', width = 33)
    labelOtrosrecursos.grid(row = 6, column = 4, sticky = 'w', columnspan = 3, padx = 20, pady = 10)
    labelalfresco = Label(miFrame, text = 'Alfresco', width = 23)
    labelalfresco.grid(row = 7, column = 4, sticky = 'w', padx = 20, pady = 2)
    alfrescoalta = Radiobutton(miFrame, text = 'ALTA', variable = alfresco, value = 1)
    alfrescoalta.grid(row = 7, column = 5, sticky = 'w', padx = 20, pady = 2)
    alfrescobaja = Radiobutton(miFrame, text = 'BAJA', variable = alfresco, value = 2)
    alfrescobaja.grid(row = 7, column = 6, sticky = 'w', padx = 20, pady = 2)
    activeDirectory(8,4,20,2)
#---------------------------Servicios de impresora usuario Profesor asociado-----------------------------
    labelfotocopiadoras = Label(miFrame, text = 'FOTOCOPIADORAS DOE:', font = ('Helvetica','14'),bg = 'SeaGreen2', width = 33)
    labelfotocopiadoras.grid(row = 10, column = 0, sticky = 'w', columnspan = 3, pady = 10)

    Checkbutton(miFrame, text = 'Usar como impresora:\nActivar en Papercut', variable = papercut,onvalue = 1,offvalue = 0).grid(row
= 11, column = 0, sticky = 'w')
    Checkbutton(miFrame, text = 'Activado en la\nfotocopiadora', variable = fotocopiadora,onvalue = 1,offvalue = 0).grid(row = 11,
column = 3, sticky = 'w')
    labelcodigofotocopiar = Label(miFrame, text = 'Código para fotocopiar:', width = 23)
    labelcodigofotocopiar.grid(row = 11, column = 1, sticky = 'nw', padx = (2,5), pady = 2)
    codigofotocopiar= Entry(miFrame, textvariable = codfoto,justify = 'center', width = 28)
    codigofotocopiar.grid(row = 11, column = 1, sticky = 'sw', padx = (2,5), pady = 2)
    labelpisofotocopia = Label(miFrame, text = 'Piso de la fotocopiadora:', width = 23)
    labelpisofotocopia.grid(row = 11, column = 2, sticky = 'nw', padx = 2, pady = 2)
    pisofotocopia= Entry(miFrame, textvariable = pisofoto,justify = 'center', width = 28)
    pisofotocopia.grid(row = 11, column = 2, sticky = 'sw', padx = 2, pady = 2)
    fechent = ''
    fechsal = ''
#----------------Usuario Profesor visitante-----------------
def visitante():
    global miFrame, miFrame2
    global
eleccion,cuadroNombre,cuadroTlf,cuadroEmail,cuadroorigen,cuadroAlta,fecha_desde,fecha_hasta,cuadrociudad,cuadropais,cuadro
ubi,cuadroUPVnet,cuadroprofres,cuadroemailprof,fecha_desde,fecha_hasta,cuadroAlta,texto,cuadroidentificacion,codigofotocopiar,
pisofotocopia,especificar
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,laboratorios,taquilla,alfresco,papercut,fotocopiadora,directo
ry,tienordenador,tieneupvnet,usuariotemporal,opcionuni
    if eleccion == 'Doctorando' or eleccion == 'Profesor asociado':
        eleccion = 'Profesor visitante'
    try:
        reiniciarEntrys()
    except:
        print()
    try:
        miFrame.destroy()
    except:
        print()
    miFrame = Frame(raiz)
    miFrame.pack(fill = 'both', expand = True)
    miFrame.config(width = 650, height = 350)
    miFrame.config(bg = 'papaya whip')
    labelvisitante = Label(miFrame, text = 'ALTA USUARIO PROFESOR VISITANTE', font = ('Helvetica','14'),bg = 'goldenrod2',
width = 33)
    labelvisitante.grid(row = 1, column = 0, sticky = 'w', columnspan = 3, pady = 12)
    eleccion = 'Profesor visitante'
    nomape(2)
    telEfono(3)
    email(3)
    uniOrigen(4)
    ciuDad(4)
    paIs(5)
    profeResp(5)
    emailProf(5)
```

```
    periodoestancia(6)
    tieneUPVNET(7)
    asignarusuariotemporal(9)
    ordenadorPersonal(10)
    fechAlta(11)
    ubi(12)
    infoAdicional(13)
    botones(14)
#---------------------------Servicios de acceso usuario Profesor visitante-----------------------------
    labelTarjetaacceso = Label(miFrame, text = 'Tarjeta acceso puertas DOE:', font = ('Helvetica','14'),bg = 'goldenrod2', width = 33)
    labelTarjetaacceso.grid(row = 1, column = 4, sticky = 'w', columnspan = 3, padx = 20, pady = 10)

    global comedor,despachos,actualizar,sem4,otro,otromotivo,codfoto,pisofoto,especificar
    Checkbutton(miFrame, text = 'Comedor', variable = comedor,onvalue = 1,offvalue = 0).grid(row = 2, column = 4, sticky = 'w', padx
= 20, pady = 2)
    Checkbutton(miFrame, text = 'Despachos Asociados', variable = despachos,onvalue = 1,offvalue = 0).grid(row = 2, column = 5,
sticky = 'w', padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Actualizador', variable = actualizar,onvalue = 1,offvalue = 0).grid(row = 2, column = 6, sticky = 'w',
padx = 20, pady = 2)
    Checkbutton(miFrame, text = 'Seminario 4', variable = sem4,onvalue = 1,offvalue = 0).grid(row = 3, column = 4, sticky = 'w', padx
= 20, pady = 2)
    Checkbutton(miFrame, text = 'Otro,especificar', variable = otro,onvalue = 1,offvalue = 0,command = check).grid(row = 3, column
= 5, sticky = 'w', padx = 20, pady = 2)
#---------------Comienzo de programa----------------
def comienzoPrograma():
    global miFrame, miFrameEditar,miFrameBorrar,miFrame2
    try:
        miFrame.destroy()
        miFrame2.destroy()
    except:
        print()
    try:
        miFrameEditar.destroy()
    except:
        print()
    try:
        miFrameBorrar.destroy()
        miFrame2.destroy()
    except:
        print()
    raiz.geometry("1270x600")
    miFrame2 = Frame(raiz)
    miFrame2.pack(fill = 'both')
    miFrame2.config(bg = 'papaya whip')
    miFrame = Frame(raiz)
    miFrame.pack(fill = 'both', expand = True)
    miFrame.config(width = 650, height = 350)
    miFrame.config(bg = 'papaya whip')

#----------------------Botones para elegir el tipo de usuario-----------------------------------
    labelUsuario = Label(miFrame2, text = 'Tipo de Usuario:',font = ('Times','12','bold'), width = 23)
    labelUsuario.grid(row = 0, column = 0, sticky = 'w')

    botonDoctorando = Button(miFrame2, command = doctorando, text = 'Doctorando', bg = 'DeepSkyBlue3', relief = 'raised', width =
23, cursor = 'hand2')
    botonDoctorando.grid(row = 0, column = 1, sticky = 'w', padx = 5, pady = 10)
    botonProfAs = Button(miFrame2, command = asociado, text = 'Profesor asociado', bg = 'SeaGreen2', relief = 'raised', width = 23,
cursor = 'hand2')
    botonProfAs.grid(row = 0, column = 2, sticky = 'w', padx = 5, pady = 10)
    botonProfVis = Button(miFrame2, command = visitante, text = 'Profesor visitante',bg = 'goldenrod2', relief = 'raised', width = 23,
cursor = 'hand2')
    botonProfVis.grid(row = 0, column = 3, sticky = 'w', padx = (5,20), pady = 10)
#---------------------------Obtener nombres de usuarios a punto de dar de baja-----------------------
    cursor.execute('select Fecha_salida_prevista from ACTIVOS')
    v = cursor.fetchall()
    miConexion.commit()
    cursor.execute('select Fecha_salida_prevista from INACTIVOS')
    x = cursor.fetchall()
    miConexion.commit()
    cursor.execute('select Nombre_y_apellidos from ACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    cursor.execute('select Nombre_y_apellidos from INACTIVOS')
    nombreyapellidos2 = cursor.fetchall()
```

```
    miConexion.commit()
    global nomusuarios, nomusuarios2
    fechaactual = datetime.now()
    longact = range(len(v))
    contador = 0
    nomusuarios = []
    nomusuarios2 = []
    for i in longact:
        st = v[i][0]
        if st != '':
            pas = st.split('-')
            year = pas[0]
            mes = pas[1]
            day = pas[2]
            daycut = day.split(' ')
            dia = daycut[0]
            fechaacomparar = str(dia+mes+year)
            formato = "%d%m%Y"
            fec = datetime.strptime(fechaacomparar, formato)
            diferencia = fec-fechaactual
            if diferencia.days <= 15 and fec !='':
                contador = contador+1
                nya = nombreyapellidos[i][0]
                nomusuarios.append(nya)
    if contador >= 1:
        messagebox.showwarning('Aviso','Hay usuarios que deben ser dados de baja en menos de 15 días de la tabla ACTIVOS')
    contador = 0
    longinac = range(len(x))
    for i in longinac:
        st2 = x[i][0]
        if st2 != '':
            pas2 = st2.split('-')
            year2 = pas2[0]
            mes2 = pas2[1]
            day2 = pas2[2]
            daycut2 = day2.split(' ')
            dia2 = daycut2[0]
            fechaacomparar2 = str(dia2+mes2+year2)
            formato = "%d%m%Y"
            fec2 = datetime.strptime(fechaacomparar2, formato)
            diferencia2 = fec2-fechaactual
            if diferencia2.days <= 15 and fec2 !='':
                contador = contador+1
                nya2 = nombreyapellidos2[i][0]
                nomusuarios2.append(nya2)
    if contador >= 1:
        messagebox.showwarning('Aviso','Hay usuarios que deben ser dados de baja en menos de 15 días de la tabla INACTIVOS')

def usuariosdardebaja():
    nuevaraiz = Tk()
    nuevaraiz.title('Usuarios a dar de baja')
    nuevaraiz.iconbitmap('logoupv.ico')
    nuevaraiz.config(bg='white')
    nuevaraiz.geometry("850x250")
    frame2 = Frame(nuevaraiz, bd=2, relief=SUNKEN)
    frame2.pack(fill = "both", expand = True)
    frame2.config(width = 650, height = 350)
    frame2.config(bg = "ghost white")
    labelacti = Label(frame2, text = 'Usuarios a dar de baja en ACTIVOS:', bg = 'dark sea green')
    labelacti.grid(row = 0, column = 0, sticky = 'w', padx = 2, pady = 2)
    fila = 0
    fila2 = 0
    for j in range(len(nomusuarios)):
        nombres = nomusuarios[j]
        fila = math.floor((j+1)/4)
        resto = (j+1)%4
        if fila == 0:
            columna = j+1
        elif resto == 0:
            jota = j
            columna = 1
        else:
            columna = j-jota
```

```
        labelnombacti = Label(frame2, text = nombres)
        labelnombacti.grid(row = fila, column = columna, sticky = 'w',padx = 2, pady = 2)
    frame3 = Frame(nuevaraiz, bd=2, relief=SUNKEN)
    frame3.pack(fill = "both", expand = True)
    frame3.config(width = 650, height = 350)
    frame3.config(bg = "white smoke")
    labelinac = Label(frame3, text = 'Usuarios a dar de baja en INACTIVOS:', bg = 'PaleGreen2')
    labelinac.grid(row = 0, column = 0, sticky = 'w',padx = 2, pady = 2)

    for h in range(len(nomusuarios2)):
        nombres2 = nomusuarios2[h]
        fila2 = math.floor((h+1)/4)
        resto2 = (h+1)%4
        if fila2 == 0:
            columna2 = h+1
        elif resto2 == 0:
            ache = h
            columna2 = 1
        else:
            columna2 = h-ache

        labelnombinacti = Label(frame3, text = nombres2)
        labelnombinacti.grid(row = fila2, column = columna2, sticky = 'w',padx = 2, pady = 2)
#---------------Nombre y apellidos---------------------
def nomape(num):
    labelNombre = Label(miFrame, text = 'Nombre y apellidos: *', width = 23)
    labelNombre.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = (0,2))
    global n,cuadroNombre
    cuadroNombre = Entry(miFrame,textvariable = n,justify = 'center',width = 60)
    cuadroNombre.grid(row = num, column = 1, sticky = 'w',columnspan = 2)
    labelCampob = Label(miFrame, text = '(*) Campo obligatorio ', bg = 'papaya whip')
    labelCampob.grid(row = num, column = 3, sticky = 'w', padx = 2, pady = 2)
#---------------Teléfono---------------------
def telEfono(num):
    labelTelefono = Label(miFrame, text = 'Teléfono:', width = 23)
    labelTelefono.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global tlf, cuadroTlf
    cuadroTlf = Entry(miFrame, textvariable = tlf,justify = 'center', width = 28)
    cuadroTlf.grid(row = num, column = 3, sticky = 'w')

#---------------Universidad de Origen---------------------
def uniOrigen(num):
    labelorigen = Label(miFrame, text = 'Universidad de origen:', width = 23)
    labelorigen.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global uniorigen,cuadroorigen
    cuadroorigen = Entry(miFrame, textvariable = uniorigen,justify = 'center', width = 28)
    cuadroorigen.grid(row = num, column = 1, sticky = 'w')
#---------------Ciudad---------------------
def ciuDad(num):
    labelciudad = Label(miFrame, text = 'Ciudad:', width = 23)
    labelciudad.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global ciudad,cuadrociudad
    cuadrociudad = Entry(miFrame, textvariable = ciudad,justify = 'center', width = 28)
    cuadrociudad.grid(row = num, column = 3, sticky = 'w')
#---------------País---------------------
def paIs(num):
    labelpais = Label(miFrame, text = 'País:', width = 23)
    labelpais.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global pais,cuadropais
    cuadropais = Entry(miFrame, textvariable = pais,justify = 'center', width = 28)
    cuadropais.grid(row = num, column = 3, sticky = 'w')
#---------------Email---------------------
def email(num):
    labelEmail = Label(miFrame, text = 'Dirección de e-mail:', width = 23)
    labelEmail.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global e,cuadroEmail
    cuadroEmail = Entry(miFrame,textvariable = e,justify = 'center', width = 28)
    cuadroEmail.grid(row = num, column = 1, sticky = 'w')
#---------------UPVNET---------------------
def upvnet(num):
    labelUPVnet = Label(miFrame, text = 'Usuario UPVNET:', width = 23)
    labelUPVnet.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global u,cuadroUPVnet
    cuadroUPVnet = Entry(miFrame, textvariable = u,justify = 'center', width = 28)
```

```
    cuadroUPVnet.grid(row = num, column = 1, sticky = 'w')
#----------------------------------------Fecha de Alta---------------------------------
def fechAlta(num):
    global alta,diasAlta,mesesAlta,anosAlta
    alta = ''
    diasAlta = ttk.Combobox(miFrame, state='readonly', width = 4)
    diasAlta['values'] = listadias
    diasAlta.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    diasAlta.set('dd')
    mesesAlta = ttk.Combobox(miFrame, state='readonly', width = 4)
    mesesAlta['values'] = listameses
    mesesAlta.grid(row = num, column = 1, padx = 2, pady = 2)
    mesesAlta.set('mm')
    anosAlta = ttk.Combobox(miFrame, state='readonly', width = 5)
    anosAlta['values'] = listaanos
    anosAlta.grid(row = num, column = 1, sticky = 'e', padx = 2, pady = 2)
    anosAlta.set('aaaa')
    labelfechadealta = Label(miFrame, text = 'Fecha de Alta:', width = 23)
    labelfechadealta.grid(row = num, column = 0, sticky = 'w', padx = (2,0), pady = 2)
#----------------------------------------Profesor Responsable--------------------------
def profeResp(num):
    global prof,cuadroprofres
    labelprofres = Label(miFrame, text = 'Profesor responsable:', width = 23)
    labelprofres.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    cuadroprofres = Entry(miFrame, textvariable = prof,justify = 'center', width = 28)
    cuadroprofres.grid(row = num, column = 1, sticky = 'w')
#--------------Email del profesor responsable---------------------
def emailProf(num):
    labelemailprof = Label(miFrame, text = 'Email del profesor:', width = 23)
    labelemailprof.grid(row = num, column = 2, sticky = 'w', padx = (10,2), pady = 2)
    global emailprof,cuadroemailprof
    cuadroemailprof = Entry(miFrame, textvariable = emailprof,justify = 'center', width = 28)
    cuadroemailprof.grid(row = num, column = 3, sticky = 'w')
#----------------------------Periodo de estancia--------------------
def periodoestancia(num):
    global fechent,fechsal,fecha_desde,fecha_hasta,diasdesde,diashasta,mesesdesde,meseshasta,anosdesde,anoshasta
    fechent = ''
    fechsal = ''
    labelperest = Label(miFrame, text = 'Periodo estancia:', width = 23)
    labelperest.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    diasdesde = ttk.Combobox(miFrame, state='readonly', width = 4)
    diasdesde['values'] = listadias
    diasdesde.grid(row = num, column = 1, sticky = 'w', padx = (10,0), pady = 2)
    diasdesde.set('dd')
    mesesdesde = ttk.Combobox(miFrame, state='readonly', width = 4)
    mesesdesde['values'] = listameses
    mesesdesde.grid(row = num, column = 1, padx = 0, pady = 2)
    mesesdesde.set('mm')
    anosdesde = ttk.Combobox(miFrame, state='readonly', width = 5)
    anosdesde['values'] = listaanos
    anosdesde.grid(row = num, column = 1, sticky = 'e', padx = (0,10), pady = 2)
    anosdesde.set('aaaa')

    diashasta = ttk.Combobox(miFrame, state='readonly', width = 4)
    diashasta['values'] = listadias
    diashasta.grid(row = num, column = 2, sticky = 'w', padx = (15,0), pady = 2)
    diashasta.set('dd')
    meseshasta = ttk.Combobox(miFrame, state='readonly', width = 4)
    meseshasta['values'] = listameses
    meseshasta.grid(row = num, column = 2, padx = 0, pady = 2)
    meseshasta.set('mm')
    anoshasta = ttk.Combobox(miFrame, state='readonly', width = 5)
    anoshasta['values'] = listaanos
    anoshasta.grid(row = num, column = 2, sticky = 'e', padx = (0,10), pady = 2)
    anoshasta.set('aaaa')
#-----------------------------Ubicacion en DOE-------------------------
def ubi(num):
    labelubi = Label(miFrame, text = 'Ubicación en DOE:', width = 23)
    labelubi.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global ubicacion,cuadroubi
    cuadroubi = Entry(miFrame, textvariable = ubicacion,justify = 'center', width = 28)
    cuadroubi.grid(row = num, column = 1, sticky = 'w')
#---------------------------Active Directory--------------------
def activeDirectory(num,col,x,y):
```

```
    global directory
    labelactivedir = Label(miFrame, text = 'Active Directory\n"Becarios-doc-imprimir"', width = 23)
    labelactivedir.grid(row = num, column = col, sticky = 'w', padx = x, pady = y)
    botonalta = Radiobutton(miFrame, text = 'ALTA', variable = directory, value = 1)
    botonalta.grid(row = num, column = col+1, sticky = 'w', padx = x, pady = y)
    botonbaja = Radiobutton(miFrame, text = 'BAJA', variable = directory, value = 2)
    botonbaja.grid(row = num, column = col+2, sticky = 'w', padx = x, pady = y)
#----------------------------Ordenador Personal--------------------
def ordenadorPersonal(num):
    global tienordenador
    labelordenador = Label(miFrame, text = 'Ordenador personal:', width = 23)
    labelordenador.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    propio = Radiobutton(miFrame, text = 'PROPIO', variable = tienordenador, value = 1)
    propio.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    doe = Radiobutton(miFrame, text = 'SOLICITA A DOE', variable = tienordenador, value = 2)
    doe.grid(row = num, column = 2, sticky = 'w', padx = 2, pady = 2)
#-----------------------------¿Tiene usuario UPVNET?----------------------
def tieneUPVNET(num):
    global tieneupvnet,cuadroidentificacion
    def tieneono():
        global u,cuadroidentificacion
        cuadroidentificacion = Entry(miFrame, textvariable = u,justify = 'center', width = 28)
        cuadroidentificacion.grid(row = num+1, column = 1, sticky = 'w')

        if tieneupvnet.get() == 1:
            Label(miFrame, text =' ', bg = 'papaya whip', width = 50).grid(row = num+1, column = 0, sticky = 'w', columnspan = 3)
            upvnet(num+1)
        else:
            labelidentificacion = Label(miFrame, text = 'Documento de identificación:', width = 23)
            labelidentificacion.grid(row = num+1, column = 0, sticky = 'w', padx = 2, pady = 2)

    labeltieneupv = Label(miFrame, text = 'Tiene usuario UPVNET:', width = 23)
    labeltieneupv.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    si = Radiobutton(miFrame, text = 'SI', variable = tieneupvnet, value = 1, command = tieneono)
    si.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    no = Radiobutton(miFrame, text = 'NO', variable = tieneupvnet, value = 2, command = tieneono)
    no.grid(row = num, column = 2, sticky = 'w', padx = 2, pady = 2)
#-----------------------------¿Asignar Usuario Temporal?----------------------
def asignarusuariotemporal(num):
    global usuariotemporal
    labelusuariotemporal = Label(miFrame, text = 'Asignar usuario temporal:', width = 23)
    labelusuariotemporal.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    si = Radiobutton(miFrame, text = 'SI', variable = usuariotemporal, value = 1)
    si.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    no = Radiobutton(miFrame, text = 'NO', variable = usuariotemporal, value = 2)
    no.grid(row = num, column = 2, sticky = 'w', padx = 2, pady = 2)
#-----------------------------Elegir universidad----------------------
def eligeuni(num):
    global opcionuni
    labeluni = Label(miFrame, text = 'Docencia en:', width = 23)
    labeluni.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    valencia = Radiobutton(miFrame, text = 'VALENCIA', variable = opcionuni, value = 1)
    valencia.grid(row = num, column = 1, sticky = 'w', padx = 2, pady = 2)
    alcoi = Radiobutton(miFrame, text = 'ALCOI', variable = opcionuni, value = 2)
    alcoi.grid(row = num, column = 2, sticky = 'w', padx = 2, pady = 2)
    gandia = Radiobutton(miFrame, text = 'GANDIA', variable = opcionuni, value = 3)
    gandia.grid(row = num, column = 3, sticky = 'w', padx = 2, pady = 2)
#---------------Información adicional----------------------
def infoAdicional(num):
    labelinfo = Label(miFrame, text = 'Información adicional:', width = 23)
    labelinfo.grid(row = num, column = 0, sticky = 'w', padx = 2, pady = 2)
    global infoadicional,texto
    texto = Entry(miFrame,textvariable = infoadicional,justify = 'center', width = 28)
    texto.grid(row = num, column = 1, sticky = 'w')
#---------------Otro motivo---------------------
def check():
    global otro,especificar,otromotivo
    especificar = Entry(miFrame, textvariable = otromotivo,justify = 'center', width = 23)
    especificar.grid(row = 3, column = 6, sticky = 'w')
    if otro.get() != 1:
        Label(miFrame, text =' ', bg = 'papaya whip', width = 23).grid(row = 3, column = 6, sticky = 'w')
##---------------Reiniciar los Entrys---------------------
def reiniciarEntrys():
```

```
    global
eleccion,cuadroNombre,cuadroTlf,cuadroEmail,cuadroorigen,diasAlta,mesesAlta,anosAlta,diasdesde,diashasta,mesesdesde,mesesha
sta,anosdesde,anoshasta,cuadrociudad,cuadropais,cuadroubi,cuadroUPVnet,cuadroprofres,cuadroemailprof,fecha_desde,fecha_hasta
,cuadroAlta,texto,cuadroidentificacion,codigofotocopiar,pisofotocopia,especificar
    global
comedor,despachos,actualizar,seminf,semtel,sem31,sem32,sem33,sem4,laboratorios,taquilla,alfresco,papercut,fotocopiadora,directo
ry,tienordenador,tieneupvnet,usuariotemporal,opcionuni,otro
    cuadroNombre.delete(0, last=len(cuadroNombre.get()))
    cuadroTlf.delete(0, last=len(cuadroTlf.get()))
    cuadroEmail.delete(0, last=len(cuadroEmail.get()))
    cuadroorigen.delete(0, last=len(cuadroorigen.get()))
    cuadrociudad.delete(0, last=len(cuadrociudad.get()))
    cuadropais.delete(0, last=len(cuadropais.get()))
    cuadroubi.delete(0, last=len(cuadroubi.get()))
    texto.delete(0, last=len(texto.get()))
    try:
        cuadroUPVnet.delete(0, last=len(cuadroUPVnet.get()))
    except:
        print()
    try:
        if tieneupvnet.get() == 2:
            cuadroidentificacion.delete(0, last=len(cuadroidentificacion.get()))
            Label(miFrame, text =' ' , bg = 'papaya whip', width = 69).grid(row = 8, column = 0, sticky = 'w', columnspan = 3)
        if tieneupvnet.get() == 1:
            Label(miFrame, text =' ' , bg = 'papaya whip', width = 69).grid(row = 8, column = 0, sticky = 'w', columnspan = 3)
        tieneupvnet.set(0)
        usuariotemporal.set(0)
    except:
        print()

    try:
        if otro.get() == 1:
            especificar.delete(0, last=len(especificar.get()))
            Label(miFrame, text =' ', bg = 'papaya whip', width = 23).grid(row = 16, column = 0, sticky = 'w')
        otro.set(0)
    except:
        print()
    try:
        cuadroprofres.delete(0, last=len(cuadroprofres.get()))
        cuadroemailprof.delete(0, last=len(cuadroemailprof.get()))
        tienordenador.set(0)
    except:
        print()
    try:
        diasAlta.set('dd')
        mesesAlta.set('mm')
        anosAlta.set('aaaa')
    except:
        print()
    try:
        diasdesde.set('dd')
        mesesdesde.set('mm')
        anosdesde.set('aaaa')
        diashasta.set('dd')
        meseshasta.set('mm')
        anoshasta.set('aaaa')
    except:
        print()

        print()
    try:
        codigofotocopiar.delete(0, last=len(codigofotocopiar.get()))
        pisofotocopia.delete(0, last=len(pisofotocopia.get()))
        fotocopiadora.set(0)
        papercut.set(0)
    except:
        print()
    if eleccion != 'Profesor asociado':
        try:
            comedor.set('0')
        except:
            print()
        despachos.set('0')
        actualizar.set('0')
```

```
    else:
        comedor.set('1')
        despachos.set('1')
        actualizar.set('1')
        laboratorios.set('1')
        taquilla.set('1')
    try:
        seminf.set(0)
        semtel.set(0)
        sem31.set(0)
        sem32.set(0)
        sem33.set(0)
    except:
        print()
    try:
        sem4.set(0)
    except:
        print()

    try:
        directory.set(0)
    except:
        print()
    try:
        alfresco.set(0)
        opcionuni.set(0)
    except:
        print()
#-------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------
#-------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------
#-------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------
#---------------------------------Crear la BBDD-------------------------------------------------------------------------------------------------------------
miConexion = sqlite3.connect('Usuarios Temporales')
cursor = miConexion.cursor()
try:
    cursor.execute('''create table ACTIVOS(
    Usuarios text,Nombre_y_apellidos text ,Email text,Teléfono integer,Universidad_de_origen text,Ciudad text,
    País text,Universidad_profesor_asociado text,Profesor_Anfitrión_DOE text,Email_profesor text,
    Fecha_Llegada datetime,Fecha_salida_prevista datetime,Fecha_salida_real datetime,
    Ubicacion_DOE text,Tipo_ordenador text,Alfresco text,Informacion_adicional text,
    Usuario_UPVNET_o_Identificación text,Asignar_usuario_temporal text,AD_Imprimir text,
    Papercut text,Activado_en_fotocopiadora text,Codigo text,Activar_fotocop_piso text,
    Comedor text,Dpch_Asociados text,Seminario_Informática text,Seminario_Teleco text,
    Seminario_3_1 text,Seminario_3_2 text,Seminario_3_3 text,Seminario_4 text,
    Actualizador text,Laboratorios text,Taquillas text,Otros_Indicar text)''')

    cursor.execute('''create table INACTIVOS(
        Usuarios text,Nombre_y_apellidos text ,Email text,Teléfono integer,Universidad_de_origen text,Ciudad text,
    País text,Universidad_profesor_asociado text,Profesor_Anfitrión_DOE text,Email_profesor text,
    Fecha_Llegada datetime,Fecha_salida_prevista datetime,Fecha_salida_real datetime,
    Ubicacion_DOE text,Tipo_ordenador text,Alfresco text,Informacion_adicional text,
    Usuario_UPVNET_o_Identificación text,Asignar_usuario_temporal text,AD_Imprimir text,
    Papercut text,Activado_en_fotocopiadora text,Codigo text,Activar_fotocop_piso text,
    Comedor text,Dpch_Asociados text,Seminario_Informática text,Seminario_Teleco text,
    Seminario_3_1 text,Seminario_3_2 text,Seminario_3_3 text,Seminario_4 text,
    Actualizador text,Laboratorios text,Taquillas text,Otros_Indicar text)''')

except:
    print()
#---------------Enviar datos a la tabla ACTIVOS---------------------
def enviaActivos():
    global
fecha_desde,fecha_hasta,diasAlta,mesesAlta,anosAlta,diasdesde,diashasta,mesesdesde,meseshasta,anosdesde,anoshasta,eleccion,n,e
,tlf,uniorigen,ciudad,pais,opcionuni,prof,emailprof,alta,fechent,fechsal,ubicacion,tienordenador,alfresco,infoadicional,u,usuariotemp
oral,directory,papercut,fotocopiadora,codfoto,pisofoto,comedor,despachos,seminf,semtel,sem31,sem32,sem33,sem4,actualizar,labor
atorios,taquilla,otromotivo
    nombre = n.get()

    telefon = tlf.get()
    email = e.get()
    origen = uniorigen.get()
```

```
ciud = ciudad.get()
p = pais.get()
ubi = ubicacion.get()
upv = u.get()
profesor = prof.get()
correoprofe = emailprof.get()
infadi= infoadicional.get()
cod = codfoto.get()
piso = pisofoto.get()
otr = otromotivo.get()
tieneordenador = tienordenador.get()
if tieneordenador == 1:
   tieneordenador = 'Propio'
elif tieneordenador == 2:
   tieneordenador = 'Del DOE'
else:
   tieneordenador = ''
universidad = opcionuni.get()
if universidad == 1:
   universidad = 'Valencia'
elif universidad == 2:
   universidad = 'Alcoi'
elif universidad == 3:
   universidad = 'Gandia'
else:
   universidad = ''

comed = comedor.get()
if comed == 1:
   comed = 'Si'
else:
   comed = 'No'

desp = despachos.get()
if desp == 1:
   desp = 'Si'
else:
   desp = 'No'

fotocop = fotocopiadora.get()
if fotocop == 1:
   fotocop = 'Si'
else:
   fotocop = 'No'

act = actualizar.get()
if act == 1:
   act= 'Si'
else:
   act = 'No'

paper = papercut.get()
if paper == 1:
   paper= 'Si'
else:
   paper = 'No'

semiinf = seminf.get()
if semiinf == 1:
   semiinf = 'Si'
else:
   semiinf = 'No'

semitel = semtel.get()
if semitel == 1:
   semitel = 'Si'
else:
   semitel = 'No'

semi31 = sem31.get()
if semi31 == 1:
   semi31 = 'Si'
else:
   semi31 = 'No'
```

```
semi32 = sem32.get()
if semi32 == 1:
   semi32 = 'Si'
else:
   semi32 = 'No'

semi33 = sem33.get()
if semi33 == 1:
   semi33 = 'Si'
else:
   semi33 = 'No'

semi4 = sem4.get()
if semi4 == 1:
   semi4 = 'Si'
else:
   semi4 = 'No'

lab = laboratorios.get()
if lab == 1:
   lab = 'Si'
else:
   lab = 'No'

taq = taquilla.get()
if taq == 1:
   taq = 'Si'
else:
   taq = 'No'

usutem = usuariotemporal.get()
if usutem == 1:
   usutem = 'Si'
elif usutem == 2:
   usutem = 'No'
else:
   usutem = ''
alf = alfresco.get()
if alf == 1:
   alf = 'Alta'
elif alf == 2:
   alf = 'Baja'
else:
   alf = ''
direc = directory.get()
if direc == 1:
   direc = 'Alta'
elif direc == 2:
   direc = 'Baja'
else:
   direc = ''

cursor.execute('select Nombre_y_apellidos from ACTIVOS')
nombreyapellidos = cursor.fetchall()
miConexion.commit()
cuenta = 0
for i in range(len(nombreyapellidos)):
   nombres = nombreyapellidos[i][0]
   if nombre == nombres:
      cuenta = cuenta+1
if nombre == '' :
   messagebox.showerror('Error','El campo Nombre y Apellidos es obligatorio')
elif cuenta > 0 and nombre != '':
   messagebox.showerror('Error','Ya existe un usuario con ese nombre')
else:
   try:
      formato = "%d%m%Y"
      try:
         if diasAlta.get() == 'dd' or mesesAlta.get() == 'mm' or anosAlta.get() == 'aaaa' :
            alta = ''
         else:
            fechaAlta = diasAlta.get()+mesesAlta.get()+anosAlta.get()
            alta = datetime.strptime(fechaAlta, formato)
```

```
            except:
                messagebox.showerror('Error en la fecha', 'Error en la fecha de Alta. ¡Inténtalo de nuevo!')
            if eleccion != 'Profesor asociado':
                try:
                    if diasdesde.get() == 'dd' or mesesdesde.get() == 'mm' or anosdesde.get() == 'aaaa' or diashasta.get() == 'dd' or
meseshasta.get() == 'mm' or anoshasta.get() == 'aaaa':
                        fechent = ''
                        fechsal = ''
                    else:
                        fechadesde = diasdesde.get()+mesesdesde.get()+anosdesde.get()
                        fechahasta = diashasta.get()+meseshasta.get()+anoshasta.get()
                        fechent = datetime.strptime(fechadesde, formato)
                        fechsal = datetime.strptime(fechahasta, formato)
                        if fechsal <= fechent:
                            messagebox.showerror('Error en la fecha', '¡La fecha final debe ser mayor o igual que la inicial!')
                except:
                    messagebox.showerror('Error en la fecha', 'Error en la/s fecha/s del periodo de estancia. ¡Inténtalo de nuevo!')


            cursor.execute('insert into ACTIVOS values(?,?,?,?,?,?,?,?,?,?,?,?,"",?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(
            eleccion,nombre,email,telefon,origen,ciud,p,universidad,profesor,correoprofe,alta,fechsal,ubi,tieneordenador,alf,
            infadi,upv,usutem,direc,paper,fotocop,cod,piso,comed,desp,semiinf,semitel,semi31,semi32,semi33,semi4,act,lab,taq,otr,))

            miConexion.commit()

            reiniciarEntrys()
        except:
            miFrame.bell()
            messagebox.showerror('Error', 'No ha sido posible enviar los datos')
#----------Envía los datos a la tabLa INACTIVOS----------
def enviaInactivos():
    global
fecha_desde,fecha_hasta,diasAlta,mesesAlta,anosAlta,diasdesde,diashasta,mesesdesde,meseshasta,anosdesde,anoshasta,eleccion,n,e
,tlf,uniorigen,ciudad,pais,opcionuni,prof,emailprof,alta,fechent,fechsal,ubicacion,tienordenador,alfresco,infoadicional,u,usuariotemp
oral,directory,papercut,fotocopiadora,codfoto,pisofoto,comedor,despachos,seminf,semtel,sem31,sem32,sem33,sem4,actualizar,labor
atorios,taquilla,otromotivo
    nombre = n.get()
    telefon = tlf.get()
    email = e.get()
    origen = uniorigen.get()
    ciud = ciudad.get()
    p = pais.get()
    ubi = ubicacion.get()
    upv = u.get()
    profesor = prof.get()
    correoprofe = emailprof.get()
    infadi= infoadicional.get()
    cod = codfoto.get()
    piso = pisofoto.get()
    otr = otromotivo.get()
    tieneordenador = tienordenador.get()
    if tieneordenador == 1:
        tieneordenador = 'Propio'
    elif tieneordenador == 2:
        tieneordenador = 'Del DOE'
    else:
        tieneordenador = ''
    universidad = opcionuni.get()
    if universidad == 1:
        universidad = 'Valencia'
    elif universidad == 2:
        universidad = 'Alcoi'
    elif universidad == 3:
        universidad = 'Gandia'
    else:
        universidad = ''
    comed = comedor.get()
    if comed == 1:
        comed = 'Si'
    else:
        comed = 'No'
    desp = despachos.get()
    if desp == 1:
        desp = 'Si'
```

```
else:
    desp = 'No'
fotocop = fotocopiadora.get()
if fotocop == 1:
    fotocop = 'Si'
else:
    fotocop = 'No'

act = actualizar.get()
if act == 1:
    act= 'Si'
else:
    act = 'No'
paper = papercut.get()
if paper == 1:
    paper= 'Si'
else:
    paper = 'No'
semiinf = seminf.get()
if semiinf == 1:
    semiinf = 'Si'
else:
    semiinf = 'No'
semitel = semtel.get()
if semitel == 1:
    semitel = 'Si'
else:
    semitel = 'No'
semi31 = sem31.get()
if semi31 == 1:
    semi31 = 'Si'
else:
    semi31 = 'No'
semi32 = sem32.get()
if semi32 == 1:
    semi32 = 'Si'
else:
    semi32 = 'No'
semi33 = sem33.get()
if semi33 == 1:
    semi33 = 'Si'
else:
    semi33 = 'No'
semi4 = sem4.get()
if semi4 == 1:
    semi4 = 'Si'
else:
    semi4 = 'No'
lab = laboratorios.get()
if lab == 1:
    lab = 'Si'
else:
    lab = 'No'

taq = taquilla.get()
if taq == 1:
    taq = 'Si'
else:
    taq = 'No'
usutem = usuariotemporal.get()
if usutem == 1:
    usutem = 'Si'
elif usutem == 2:
    usutem = 'No'
else:
    usutem = ''
alf = alfresco.get()
if alf == 1:
    alf = 'Alta'
elif alf == 2:
    alf = 'Baja'
else:
    alf = ''
direc = directory.get()
```

```
    if direc == 1:
        direc = 'Alta'
    elif direc == 2:
        direc = 'Baja'
    else:
        direc = ''
    cursor.execute('select Nombre_y_apellidos from INACTIVOS')
    nombreyapellidos = cursor.fetchall()
    miConexion.commit()
    cuenta = 0
    for i in range(len(nombreyapellidos)):
        nombres = nombreyapellidos[i][0]
        if nombre == nombres:
            cuenta = cuenta+1
    if nombre == '' :
        messagebox.showerror('Error','El campo Nombre y Apellidos es obligatorio')
    elif cuenta > 0 and nombre != '':
        messagebox.showerror('Error','Ya existe un usuario con ese nombre')
    else:
        try:
            formato = "%d%m%Y"
            try:
                if diasAlta.get() == 'dd' or mesesAlta.get() == 'mm' or anosAlta.get() == 'aaaa' :
                    alta = ''
                else:
                    fechaAlta = diasAlta.get()+mesesAlta.get()+anosAlta.get()
                    alta = datetime.strptime(fechaAlta, formato)
            except:
                messagebox.showerror('Error en la fecha', 'Error en la fecha de Alta. ¡Inténtalo de nuevo!')
            if eleccion != 'Profesor asociado':
                try:
                    if diasdesde.get() == 'dd' or mesesdesde.get() == 'mm' or anosdesde.get() == 'aaaa' or diashasta.get() == 'dd' or
meseshasta.get() == 'mm' or anoshasta.get() == 'aaaa':
                        fechent = ''
                        fechsal = ''
                    else:
                        fechadesde = diasdesde.get()+mesesdesde.get()+anosdesde.get()
                        fechahasta = diashasta.get()+meseshasta.get()+anoshasta.get()
                        fechent = datetime.strptime(fechadesde, formato)
                        fechsal = datetime.strptime(fechahasta, formato)
                        if fechsal <= fechent:
                            messagebox.showerror('Error en la fecha', '¡La fecha final debe ser mayor o igual que la inicial!')
                except:
                    messagebox.showerror('Error en la fecha', 'Error en la/s fecha/s del periodo de estancia. ¡Inténtalo de nuevo!')

            cursor.execute('insert into INACTIVOS values(?,?,?,?,?,?,?,?,?,?,?,?,?,"",?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(
            eleccion,nombre,email,telefon,origen,ciud,p,universidad,profesor,correoprofe,alta,fechsal,ubi,tieneordenador,alf,
            infadi,upv,usutem,direc,paper,fotocop,cod,piso,comed,desp,semiinf,semitel,semi31,semi32,semi33,semi4,act,lab,taq,otr,))
            miConexion.commit()
            reiniciarEntrys()
        except:
            miFrame.bell()
            messagebox.showerror('Error', 'No ha sido posible enviar los datos')
#----------------------------------------------Editar Base de datos-------------------------------
def editarBD():
    global miFrame, miFrameEditar,miFrameBorrar,miFrame2
    try:
        miFrameEditar.destroy()
    except:
        print()
    try:
        miFrame.destroy()
        miFrame2.destroy()
    except:
        print()
    try:
        miFrameBorrar.destroy()
    except:
        print()
    miFrameEditar = Frame(raiz)
    miFrameEditar.config(width = 650, height = 350)
    miFrameEditar.config(bg = 'DarkOliveGreen1')
    miFrameEditar.pack(fill = 'both', expand = True)
    raiz.geometry("850x350")
```

```
    def eligeTabla():
        actoinact = tablaaeditar.get()
        if actoinact == 'Activos':
            editarActivos()
        elif actoinact == 'Inactivos':
            editarInactivos()
    labelEditar = Label(miFrameEditar, text = 'EDITAR BASE DE DATOS', font = ('Helvetica','14'),bg = 'wheat1')
    labelEditar.grid(row = 0, column = 0, sticky = 'w', columnspan = 2, pady = 10)
    labeltabla = Label(miFrameEditar, text = 'Tabla a editar:', width = 28)
    labeltabla.grid(row = 1, column = 0, sticky = 'w', padx = 2, pady = 2)
    tablaaeditar = ttk.Combobox(miFrameEditar, state='readonly', width = 28)
    tablaaeditar['values'] = ['Activos','Inactivos']
    tablaaeditar.grid(row = 1, column = 1, sticky = 'w', padx = 2, pady = 2)
    selecboton = Button(miFrameEditar, command = eligeTabla, text = 'Seleccionar', relief = 'raised' , bg = 'khaki1', cursor = 'hand2')
    selecboton.grid(row = 1, column = 2, sticky = 'w', padx = 25, pady = 2)
    def editarActivos():
        global nom,nuevo,elec,quemodificar,quienmodificar,insertnuevo
        cursor.execute('select Nombre_y_apellidos from ACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quienmodificar['values'] = nombap
        quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)
        nombreusuario = Label(miFrameEditar, text = 'Nombre del usuario a modificar:', width = 28)
        nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
        campoamodificar = Label(miFrameEditar, text = 'Campo a modificar:', width = 28)
        campoamodificar.grid(row = 3, column = 0, sticky = 'w', padx = 2, pady = 2)
        quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quemodificar['values']=
['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
'Profesor_Anfitrión_DOE','Email_profesor','Fecha_Llegada','Fecha_salida_prevista','Fecha_salida_real',
'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación'
'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
        quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
        quemodificar.bind("<<ComboboxSelected>>", nuevaseleccion)

    def nuevaseleccion(event):
        global nuevo,diasEditarAct,mesesEditarAct,anosEditarAct,insertnuevo
        labelnuevo = Label(miFrameEditar, text = 'Nuevo valor:', width = 28)
        labelnuevo.grid(row = 4, column = 0, sticky = 'w', padx = 2, pady = 2)
        try:
            insertnuevo.destroy()
        except:
            print()
        try:
            diasEditarAct.destroy()
            mesesEditarAct.destroy()
            anosEditarAct.destroy()
        except:
            print()
        if quemodificar.get() == 'Fecha_Llegada' or quemodificar.get() == 'Fecha_salida_prevista' or quemodificar.get() ==
'Fecha_salida_real':
            diasEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
            diasEditarAct['values'] = listadias
            diasEditarAct.grid(row = 4, column = 1, sticky = 'w', padx = 2, pady = 2)
            diasEditarAct.set('dd')
            mesesEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
            mesesEditarAct['values'] = listameses
            mesesEditarAct.grid(row = 4, column = 1, padx = 2, pady = 2)
            mesesEditarAct.set('mm')
            anosEditarAct = ttk.Combobox(miFrameEditar, state='readonly', width = 5)
            anosEditarAct['values'] = listaanos
            anosEditarAct.grid(row = 4, column = 1, sticky = 'e', padx = 2, pady = 2)
            anosEditarAct.set('aaaa')
        else:
            insertnuevo = Entry(miFrameEditar, textvariable = nuevo,justify = 'center', width = 50)
            insertnuevo.grid(row = 4, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
```

```
    boton = Button(miFrameEditar, command = Aceptar, text = 'Aplicar', cursor = 'hand2')
    boton.grid(row = 5, column = 3, sticky = 'w', padx = 5, pady = 20)

def Aceptar():
    global nuevo,elec,nom,quemodificar,quienmodificar,insertnuevo,diasEditarAct,mesesEditarAct,anosEditarAct
    new = nuevo.get()
    try:
        newfechas = diasEditarAct.get()+mesesEditarAct.get()+anosEditarAct.get()
    except:
        print()
    try:
        insertnuevo.delete(0, last = len(new))
    except:
        print()
    try:
        diasEditarAct.set('dd')
        mesesEditarAct.set('mm')
        anosEditarAct.set('aaaa')
    except:
        print()
    elec = quemodificar.get()
    nom = quienmodificar.get()
    if elec=='Usuarios':
        cursor.execute('update ACTIVOS set Usuarios = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Nombre_y_apellidos':
        cursor.execute('update ACTIVOS set Nombre_y_apellidos = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Email':
        cursor.execute('update ACTIVOS set Email = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Teléfono':
        cursor.execute('update ACTIVOS set Teléfono = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Universidad_de_origen':
        cursor.execute('update ACTIVOS set Universidad_de_origen = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Ciudad':
        cursor.execute('update ACTIVOS set Ciudad = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='País':
        cursor.execute('update ACTIVOS set País = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Universidad_profesor_asociado':
        cursor.execute('update ACTIVOS set Universidad_profesor_asociado = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Profesor_Anfitrión_DOE':
        cursor.execute('update ACTIVOS set Profesor_Anfitrión_DOE = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Email_profesor':
        cursor.execute('update ACTIVOS set Email_profesor = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Fecha_Llegada':
        formato = "%d%m%Y"
        try:
            new = datetime.strptime(newfechas, formato)
            cursor.execute('update ACTIVOS set Fecha_Llegada = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        except:
            messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')

    if elec=='Fecha_salida_prevista':
        formato = "%d%m%Y"
        try:
            new = datetime.strptime(newfechas, formato)
            cursor.execute('update ACTIVOS set Fecha_salida_prevista = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        except:
            messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')
    if elec=='Fecha_salida_real':
        formato = "%d%m%Y"
        try:
            new = datetime.strptime(newfechas, formato)
```

```
            cursor.execute('update ACTIVOS set Fecha_salida_real = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
          except:
            messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')
      if elec=='Ubicacion_DOE':
        cursor.execute('update ACTIVOS set Ubicacion_DOE = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Tipo_ordenador':
        cursor.execute('update ACTIVOS set Tipo_ordenador = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Alfresco':
        cursor.execute('update ACTIVOS set Alfresco = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Informacion_adicional':
        cursor.execute('update ACTIVOS set Informacion_adicional = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Usuario_UPVNET_o_Identificación':
        cursor.execute('update ACTIVOS set Usuario_UPVNET_o_Identificación = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Asignar_usuario_temporal':
        cursor.execute('update ACTIVOS set Asignar_usuario_temporal = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='AD_Imprimir':
        cursor.execute('update ACTIVOS set AD_Imprimir = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Papercut':
        cursor.execute('update ACTIVOS set Papercut = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Activado_en_fotocopiadora':
        cursor.execute('update ACTIVOS set Activado_en_fotocopiadora = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Codigo':
        cursor.execute('update ACTIVOS set Codigo = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Activar_fotocop_piso':
        cursor.execute('update ACTIVOS set Activar_fotocop_piso = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Comedor':
        cursor.execute('update ACTIVOS set Comedor = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Dpch_Asociados':
        cursor.execute('update ACTIVOS set Dpch_Asociados = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_Informática':
        cursor.execute('update ACTIVOS set Seminario_Informática = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_Teleco':
        cursor.execute('update ACTIVOS set Seminario_Teleco = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_3_1':
        cursor.execute('update ACTIVOS set Seminario_3_1 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_3_2':
        cursor.execute('update ACTIVOS set Seminario_3_2 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_3_3':
        cursor.execute('update ACTIVOS set Seminario_3_3 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Seminario_4':
        cursor.execute('update ACTIVOS set Seminario_4 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Actualizador':
        cursor.execute('update ACTIVOS set Actualizador = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Laboratorios':
        cursor.execute('update ACTIVOS set Laboratorios = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Taquillas':
        cursor.execute('update ACTIVOS set Taquillas = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
      if elec=='Otros_Indicar':
        cursor.execute('update ACTIVOS set Otros_Indicar = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
```

```
        cursor.execute('select Nombre_y_apellidos from ACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)

        quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quienmodificar['values'] = nombap
        quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)

        quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quemodificar['values']=
['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
'Profesor_Anfitrión_DOE','Email_profesor','Fecha_Llegada','Fecha_salida_prevista','Fecha_salida_real',
'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación'
'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
        quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
    def editarInactivos():
        global nom,nuevo,elec,quemodificar,quienmodificar,insertnuevo
        cursor.execute('select Nombre_y_apellidos from INACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)

        quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quienmodificar['values'] = nombap
        quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)
        nombreusuario = Label(miFrameEditar, text = 'Nombre del usuario a modificar:', width = 28)
        nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
        campoamodificar = Label(miFrameEditar, text = 'Campo a modificar:', width = 28)
        campoamodificar.grid(row = 3, column = 0, sticky = 'w', padx = 2, pady = 2)
        quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quemodificar['values']=
['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
'Profesor_Anfitrión_DOE','Email_profesor','Fecha_Llegada','Fecha_salida_prevista','Fecha_salida_real',
'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación'
'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
        quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)
        quemodificar.bind("<<ComboboxSelected>>", nuevaseleccionInac)

    def nuevaseleccionInac(event):
        global nuevo,diasEditarInac,mesesEditarInac,anosEditarInac,insertnuevoInac
        labelnuevo = Label(miFrameEditar, text = 'Nuevo valor:', width = 28)
        labelnuevo.grid(row = 4, column = 0, sticky = 'w', padx = 2, pady = 2)
        try:
            insertnuevoInac.destroy()
        except:
            print()
        try:
            diasEditarInac.destroy()
            mesesEditarInac.destroy()
            anosEditarInac.destroy()
        except:
            print()
        if quemodificar.get() == 'Fecha_Llegada' or quemodificar.get() == 'Fecha_salida_prevista' or quemodificar.get() ==
'Fecha_salida_real':
            diasEditarInac = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
            diasEditarInac['values'] = listadias
            diasEditarInac.grid(row = 4, column = 1, sticky = 'w', padx = 2, pady = 2)
            diasEditarInac.set('dd')
            mesesEditarInac = ttk.Combobox(miFrameEditar, state='readonly', width = 4)
            mesesEditarInac['values'] = listameses
            mesesEditarInac.grid(row = 4, column = 1, padx = 2, pady = 2)
            mesesEditarInac.set('mm')
```

```
        anosEditarInac = ttk.Combobox(miFrameEditar, state='readonly', width = 5)
        anosEditarInac['values'] = listaanos
        anosEditarInac.grid(row = 4, column = 1, sticky = 'e', padx = 2, pady = 2)
        anosEditarInac.set('aaaa')
    else:
        insertnuevoInac = Entry(miFrameEditar, textvariable = nuevo,justify = 'center', width = 50)
        insertnuevoInac.grid(row = 4, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)

    boton = Button(miFrameEditar, command = AceptarInac, text = 'Aplicar', cursor = 'hand2')
    boton.grid(row = 5, column = 3, sticky = 'w', padx = 5, pady = 20)
def AceptarInac():
    global nuevo,elec,nom,quemodificar,quienmodificar,insertnuevo,diasEditarInac,mesesEditarInac,anosEditarInac
    new = nuevo.get()
    try:
        newfechas = diasEditarInac.get()+mesesEditarInac.get()+anosEditarInac.get()
    except:
        print()
    try:
        insertnuevo.delete(0, last = len(new))
    except:
        print()
    try:
        diasEditarInac.set('dd')
        mesesEditarInac.set('mm')
        anosEditarInac.set('aaaa')
    except:
        print()
    elec = quemodificar.get()
    nom = quienmodificar.get()
    if elec=='Usuarios':
        cursor.execute('update INACTIVOS set Usuarios = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Nombre_y_apellidos':
        cursor.execute('update INACTIVOS set Nombre_y_apellidos = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Email':
        cursor.execute('update INACTIVOS set Email = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Teléfono':
        cursor.execute('update INACTIVOS set Teléfono = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Universidad_de_origen':
        cursor.execute('update INACTIVOS set Universidad_de_origen = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Ciudad':
        cursor.execute('update INACTIVOS set Ciudad = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='País':
        cursor.execute('update INACTIVOS set País = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Universidad_profesor_asociado':
        cursor.execute('update INACTIVOS set Universidad_profesor_asociado = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Profesor_Anfitrión_DOE':
        cursor.execute('update INACTIVOS set Profesor_Anfitrión_DOE = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Email_profesor':
        cursor.execute('update INACTIVOS set Email_profesor = (?) where Nombre_y_apellidos = (?)',(new,nom,))
        miConexion.commit()
    if elec=='Fecha_Llegada':
        formato = "%d%m%Y"
        try:
            new = datetime.strptime(newfechas, formato)
            cursor.execute('update INACTIVOS set Fecha_Llegada = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        except:
            messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')

    if elec=='Fecha_salida_prevista':
        formato = "%d%m%Y"
        try:
            new = datetime.strptime(newfechas, formato)
            cursor.execute('update INACTIVOS set Fecha_salida_prevista = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
```

```
            except:
                messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')

        if elec=='Fecha_salida_real':
            formato = "%d%m%Y"
            try:
                new = datetime.strptime(newfechas, formato)
                cursor.execute('update INACTIVOS set Fecha_salida_real = (?) where Nombre_y_apellidos = (?)',(new,nom,))
                miConexion.commit()
            except:
                messagebox.showerror('Error en la fecha', 'Error en la fecha. ¡Inténtalo de nuevo!')
        if elec=='Ubicacion_DOE':
            cursor.execute('update INACTIVOS set Ubicacion_DOE = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Tipo_ordenador':
            cursor.execute('update INACTIVOS set Tipo_ordenador = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Alfresco':
            cursor.execute('update INACTIVOS set Alfresco = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Informacion_adicional':
            cursor.execute('update INACTIVOS set Informacion_adicional = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Usuario_UPVNET_o_Identificación':
            cursor.execute('update INACTIVOS set Usuario_UPVNET_o_Identificación = (?) where Nombre_y_apellidos =
(?)',(new,nom,))
            miConexion.commit()
        if elec=='Asignar_usuario_temporal':
            cursor.execute('update INACTIVOS set Asignar_usuario_temporal = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='AD_Imprimir':
            cursor.execute('update INACTIVOS set AD_Imprimir = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Papercut':
            cursor.execute('update INACTIVOS set Papercut = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Activado_en_fotocopiadora':
            cursor.execute('update INACTIVOS set Activado_en_fotocopiadora = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Codigo':
            cursor.execute('update INACTIVOS set Codigo = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Activar_fotocop_piso':
            cursor.execute('update INACTIVOS set Activar_fotocop_piso = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Comedor':
            cursor.execute('update INACTIVOS set Comedor = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Dpch_Asociados':
            cursor.execute('update INACTIVOS set Dpch_Asociados = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_Informática':
            cursor.execute('update INACTIVOS set Seminario_Informática = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_Teleco':
            cursor.execute('update INACTIVOS set Seminario_Teleco = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_3_1':
            cursor.execute('update INACTIVOS set Seminario_3_1 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_3_2':
            cursor.execute('update INACTIVOS set Seminario_3_2 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_3_3':
            cursor.execute('update INACTIVOS set Seminario_3_3 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Seminario_4':
            cursor.execute('update INACTIVOS set Seminario_4 = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Actualizador':
            cursor.execute('update INACTIVOS set Actualizador = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Laboratorios':
            cursor.execute('update INACTIVOS set Laboratorios = (?) where Nombre_y_apellidos = (?)',(new,nom,))
```

```
            miConexion.commit()
        if elec=='Taquillas':
            cursor.execute('update INACTIVOS set Taquillas = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()
        if elec=='Otros_Indicar':
            cursor.execute('update INACTIVOS set Otros_Indicar = (?) where Nombre_y_apellidos = (?)',(new,nom,))
            miConexion.commit()

        cursor.execute('select Nombre_y_apellidos from INACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quienmodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quienmodificar['values'] = nombap
        quienmodificar.grid(row = 2, column = 1, sticky = 'w', columnspan = 3, padx = 2, pady = 2)

        quemodificar = ttk.Combobox(miFrameEditar, state='readonly', width = 50)
        quemodificar['values']=
['Usuarios','Nombre_y_apellidos','Email','Teléfono','Universidad_de_origen','Ciudad','País','Universidad_profesor_asociado',
'Profesor_Anfitrión_DOE','Email_profesor','Fecha_Llegada','Fecha_salida_prevista','Fecha_salida_real',
'Ubicacion_DOE','Tipo_ordenador','Alfresco','Informacion_adicional','Usuario_UPVNET_o_Identificación'
'Asignar_usuario_temporal','AD_Imprimir','Papercut','Activado_en_fotocopiadora','Codigo','Activar_fotocop_piso','Comedor',
'Dpch_Asociados' ,'Seminario_Informática' ,'Seminario_Teleco' ,'Seminario_3_1' ,'Seminario_3_2','Seminario_3_3',
'Seminario_4','Actualizador','Laboratorios','Taquillas','Otros_Indicar']
        quemodificar.grid(row = 3, column = 1, sticky = 'w', columnspan=3, padx = 2, pady = 2)
#-----------------------------------------------Borrar registros de la BBDD-----------------------------------------
def borrar():
    global miFrame, miFrameEditar,miFrameBorrar,miFrame2
    try:
        miFrameBorrar.destroy()
    except:
        print()
    try:
        miFrame.destroy()
        miFrame2.destroy()
    except:
        print()
    try:
        miFrameEditar.destroy()
    except:
        print()
    miFrameBorrar = Frame(raiz)
    miFrameBorrar.pack(fill = 'both', expand = True)
    miFrameBorrar.config(width = 650, height = 350)
    miFrameBorrar.config(bg = 'coral1')
    raiz.geometry("850x350")

    def eligeTabla():
        actoinact = tablaaeliminar.get()
        if actoinact == 'Activos':
            eliminarActivos()
        elif actoinact == 'Inactivos':
            eliminarInactivos()
    labelEliminar = Label(miFrameBorrar, text = 'ELIMINAR REGISTRO', font = ('Helvetica','14'),bg = 'wheat1')
    labelEliminar.grid(row = 0, column = 0, sticky = 'w', columnspan = 2, pady = 10)
    labeltabla = Label(miFrameBorrar, text = 'Tabla en la que está ubicado el registro a eliminar:', width = 38)
    labeltabla.grid(row = 1, column = 0, sticky = 'w', padx = 2, pady = 2)

    tablaaeliminar = ttk.Combobox(miFrameBorrar, state='readonly', width = 28)
    tablaaeliminar['values'] = ['Activos','Inactivos']
    tablaaeliminar.grid(row = 1, column = 1, sticky = 'w', padx = 2, pady = 2)
    selecboton = Button(miFrameBorrar, command = eligeTabla, text = 'Seleccionar', cursor = 'hand2')
    selecboton.grid(row = 1, column = 2, sticky = 'w', padx = 25, pady = 2)
    def eliminarActivos():
        global quieneliminaract
        nombreusuario = Label(miFrameBorrar, text = 'Nombre del usuario a eliminar:', width = 38)
        nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
        cursor.execute('select Nombre_y_apellidos from ACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
```

```
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quieneliminaract  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
        quieneliminaract ['values'] = nombap
        quieneliminaract .grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)

        selecboton = Button(miFrameBorrar, command = EliminarAct, text = 'Eliminar', bg = 'PaleTurquoise2', cursor = 'hand2')
        selecboton.grid(row = 3, column = 3, sticky = 'w', padx = 5, pady = 20)
    def eliminarInactivos():
        global quieneliminarinact
        nombreusuario = Label(miFrameBorrar, text = 'Nombre del usuario a eliminar:', width = 38)
        nombreusuario.grid(row = 2, column = 0, sticky = 'w', padx = 2, pady = 2)
        cursor.execute('select Nombre_y_apellidos from INACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quieneliminarinact  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
        quieneliminarinact ['values'] = nombap
        quieneliminarinact .grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)
        selecboton = Button(miFrameBorrar, command = EliminarInact, text = 'Eliminar', bg = 'PaleTurquoise2', cursor = 'hand2')
        selecboton.grid(row = 3, column = 3, sticky = 'w', padx = 5, pady = 20)
    def EliminarAct():
        global quieneliminaract
        nom = quieneliminaract .get()
        cursor.execute('delete from ACTIVOS where Nombre_y_apellidos = (?)',(nom,))
        miConexion.commit()
        cursor.execute('select Nombre_y_apellidos from ACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quieneliminaract  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
        quieneliminaract ['values'] = nombap
        quieneliminaract.grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)
    def EliminarInact():
        global quieneliminarinact
        nom = quieneliminarinact.get()
        cursor.execute('delete from INACTIVOS where Nombre_y_apellidos = (?)',(nom,))
        miConexion.commit()
        cursor.execute('select Nombre_y_apellidos from INACTIVOS')
        nombreyapellidos = cursor.fetchall()
        miConexion.commit()
        nombap = []
        for i in range(len(nombreyapellidos)):
            nombres = nombreyapellidos[i][0]
            nombap.append(nombres)
        quieneliminarinact  = ttk.Combobox(miFrameBorrar, state='readonly', width = 50)
        quieneliminarinact ['values'] = nombap
        quieneliminarinact.grid(row = 2, column = 1, sticky = 'w',columnspan = 3, padx = 2, pady = 2)
#---------------------------------------------Salir-----------------------------------
def Salir():
    valor = messagebox.askquestion('Salir','¿Está seguro de que desea salir del programa?')
    if valor == 'yes':
        miConexion.close()
        raiz.destroy()

def volverFormulario():
    comienzoPrograma()
#-----------------------------------------------Barra de Menu----------------------------
barraMenu = Menu(raiz)
raiz.config(menu = barraMenu)
dardebaja = Menu(barraMenu, tearoff = 0)
barraMenu.add_command(label = 'Formulario', command = volverFormulario)
barraMenu.add_command(label = 'Editar BBDD', command = editarBD)
barraMenu.add_command(label = 'Eliminar registros', command = borrar)
barraMenu.add_cascade(label = "Ver", menu = dardebaja)
dardebaja.add_command(label = 'Usuarios a punto de dar de baja...', command = usuariosdardebaja)
barraMenu.add_command(label = 'Salir', command = Salir)
```

```
#-------------------------------------------------------------------------------------------
comienzoPrograma()
#----------------------------------Botones para enviar los datos-------------------------
def botones(num):
    almacenar = Label(miFrame, text = 'Almacenar los datos en:', font = ('Helvetica','13'), bg = 'tomato2')
    almacenar.grid(row = num, column = 0, sticky = 'we', columnspan = 2, padx = 15, pady = 30)
    botonAct = Button(miFrame, text = 'ACTIVOS', command = enviaActivos, cursor = 'hand2')
    botonAct.grid(row = num, column = 2, sticky = 'w', pady = 30)
    botonInac = Button(miFrame, text = 'INACTIVOS', command = enviaInactivos, cursor = 'hand2')
    botonInac.grid(row = num, column = 3, sticky = 'w', pady = 30)

raiz.mainloop()
```