

**CURSO PYTHON 3 – NIVEL INICIAL**  
**ENTREGA FINAL**

---

**MANUAL DE LA APLICACIÓN**  
**DOCUMENTACION TECNICA**

**INDICE**

1. Descripción general.
2. Pantalla principal – Sus componentes.
3. Manual de uso.
  - 3a. Alta de un registro.
  - 3b. Baja de un registro.
  - 3c. Modificación de un registro.
  - 3d. Limpieza de las cajas de entrada.
  - 3e. Filtro por nombre de obra.
  - 3f. Selección en listado.
  - 3g. Barra de menú.
4. Módulos de la aplicación.
5. Detalle de las funciones del modelo.
  - 5a. Funciones para el manejo de la base de datos.
  - 5b. Funciones de ABM.
  - 5c. Funciones de consulta y treeview.
  - 5d. Función de cierre de la aplicación.
  - 5e. Funciones de manipulación de datos.
  - 5f. Declaración de la variable “info”.
6. Vista y control.
  - 6a. Raíz de la ventana.
  - 6b. Frame de menú.
  - 6c. Frame de datos.
  - 6d. Frame de treeview.
  - 6e. Label de status.

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

### 1. Descripción general

Se trata de una aplicación para el manejo de una nómina de empleados.

La visualización de la misma será con TKinter, que por medio de sus widgets podremos hacer las altas, bajas, modificaciones y consultas (CRUD). La información a manipular será guardada en una base de datos a través de SQLite3.

### 2. Pantalla principal – Sus componentes



A Área de menú de comandos.

- A.1 Botón para dar de alta al registro.
- A.2 Botón para la baja del registro previamente seleccionado.
- A.3 Botón para la modificación del registro previamente seleccionado.
- A.4 Botón para limpiar los campos de ingreso de datos.
- A.5 Botón para salir de la aplicación.

B Área para manejo de datos.

- B.1 DNI (será controlado el tipo y longitud de dato) dato obligatorio.
- B.2 CUIL (será controlado el tipo y longitud de dato) dato obligatorio.
- B.3 Nombres de la persona, dato obligatorio.
- B.4 Apellidos de la persona, dato obligatorio.
- B.5 Domicilio en el cual reside actualmente.
- B.6 Fecha de nacimiento, se puede ingresar directamente o con un almanaque.

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

- B.7 Fecha de alta, por defecto aparece la fecha actual, pero se puede modificar con la misma metodología que el punto anterior.
- B.8 Obra a la cual fue asignado a trabajar.
- B.9 Seguro de accidentes personales o A.R.T.
- B.10 El jornal que percibe por día de trabajo.
- B.11 Botón de búsqueda.

### C Área Listado

- C.1 Ingresar nombre de obra por la que se quiere filtrar el listado.
- C.2 Botón para aplicar el filtro.
- C.3 Listado de todas las obras o el contenido filtrado.

### D Área de notificación o status de las actividades.

## 3. Manual de uso

### 3.a. Alta de un registro:

Para poder dar de alta a un empleado es necesario completar todos los campos, en especial los obligatorios (DNI (B.1), CUIL (B.2), Nombres (B.3) y Apellidos (B.4)) Para el DNI es necesario que sean solo números y una cantidad mínima de dígitos de 7 y máxima de 8 sin colocar puntos. Para CUIL son 11 dígitos y sin colocar los guiones.

Para las fechas se pueden ingresar directamente o a través de los almanaques que se despliegan.

Una vez completados todos los datos se debe dar al botón de ALTA (A.1) y será agregado al listado de abajo (C.3) y grabado en la base de datos. En caso de errores será alertado con una ventana emergente y en todos los casos se le informará sobre el proceso en el área de notificación.

ID	DNI	NOMBRES	APELLIDOS	OBRA ASIGNADA	JORNAL (\$)
1	3344555	Jose	Perez	Casa 1	18000.0
2	40700900	Pedro	Gonzalez	Casa 2	20000.0
3	35653356	Jacinto	Garcia	Casa 1	19000.0
4	39123321	Mario	Fernandez	Casa 2	21000.0
5	28001133	Martin	Gomez	Casa 3	23000.0

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

### 3.b. Baja de un registro:

Hay dos caminos distintos para dar de baja a algún registro.

1. Una manera es seleccionando el mismo desde el listado en la parte inferior y luego se completarán todos los campos (B) con los datos correspondientes a ese registro.
2. La otra alternativa es ingresar el número de DNI en el campo (B.1) con dicho nombre y presionar el botón de Buscar (B.11). Este traerá a los campos los datos correspondientes de ese registro. En caso que no exista en la base de datos ese número de DNI se dará una advertencia.

En ambos métodos luego se debe presionar el botón de BAJA (A.2) se consultará si está seguro de ejecutar la sentencia y con la afirmación se borrará la información en el listado de abajo y de la base de datos. En caso de errores será alertado con una ventana emergente y en todos los casos se le informará sobre el proceso en el área de notificación.

### OPCION 1:

The screenshot shows a Windows application window titled "Evaluacion final - Python inicial". The main title bar says "GESTION DE NOMINA DE EMPLEADOS". On the left is a vertical menu with buttons for ALTA, BAJA (highlighted in orange), MODIFICACION, LIMPIAR, and SALIR. Number 2 is placed over the BAJA button. The right side contains two main sections: "INFORMACION DEL EMPLEADO" and "CAMPOS BLOQUEADOS". In the "INFORMACION DEL EMPLEADO" section, fields include D.N.I. (35653356), Buscar, C.U.I.L. (20356533567), NOMBRES (Jacinto), APELLIDOS (Garcia), DOMICILIO (La Callecita), FECHA DE NAC. (15-04-1990), OBRA ASIGNADA (Casa 1), ART o SEGURO (Seguridad Total), and JORNAL \$. Below these is a note: "\* En el campo DNI y CUIL solo ingrese números sin ',' o '-'". In the "CAMPOS BLOQUEADOS" section, there is a green bar with the text "Puede modificar o dar de baja al registro.". At the bottom of the application window, number 1 is placed over the first row of a table listing employees. The table has columns: ID, DNI, NOMBRES, APELLIDOS, OBRA ASIGNADA, and JORNAL (\$). The data in the table is as follows:

ID	DNI	NOMBRES	APELLIDOS	OBRA ASIGNADA	JORNAL (\$)
1	33444555	Jose	Perez	Casa 1	18000.0
2	40700900	Pedro	Gonzalez	Casa 2	20000.0
3	35653356	Jacinto	Garcia	Casa 1	19000.0
4	39123321	Mario	Fernandez	Casa 2	21000.0
5	28001133	Martin	Gomez	Casa 3	23000.0
6	44959151	Maria Dolores	Snachez	Casa 3	22000.0

### OPCION 2:

The screenshot shows the same application window as the previous one. The menu on the left has BAJA highlighted in orange, and number 3 is placed over it. The right side shows the "INFORMACION DEL EMPLEADO" and "CAMPOS BLOQUEADOS" sections. The "INFORMACION DEL EMPLEADO" section includes fields for D.N.I. (139123321), Buscar, C.U.I.L. (23391233218), NOMBRES (Mario), APELLIDOS (Fernandez), DOMICILIO (Maza), FECHA DE NAC. (18-10-1993), OBRA ASIGNADA (Casa 2), ART o SEGURO (Seguridad Total), and JORNAL \$. Below these fields is the note: "\* En el campo DNI y CUIL solo ingrese números sin ',' o '-'". The "CAMPOS BLOQUEADOS" section is present but empty. At the bottom of the application window, number 1 is placed over the first row of a table listing employees, which is identical to the one in the previous screenshot.

## CURSO PYTHON 3 – NIVEL INICIAL

### ENTREGA FINAL

#### 3.c. Modificación de un registro:

Los procedimientos para la modificación de algún dato correspondiente a un registro son:

1. Una manera es seleccionando el mismo desde el listado en la parte inferior y luego se completarán todos los campos (B) con los datos correspondientes a ese registro.
2. La otra alternativa es ingresar el número de DNI en el campo (B.1) con dicho nombre y presionar el botón de Buscar (B.11). Este traerá a los campos los datos correspondientes de ese registro. En caso que no exista en la base de datos ese número de DNI se dará una advertencia.

En ambos métodos se bloquearán los campos de DNI (B.1) y CUIL (B.2), los cuales no podrán ser modificados. Luego se debe presionar el botón de MODIFICAR (A.3) se consultará si está seguro de ejecutar la sentencia y con la afirmación se actualizará la información del listado de abajo y de la base de datos. En caso de errores será alertado con una ventana emergente y en todos los casos se le informará sobre el proceso en el área de notificación.

#### OPCION 1:

**GESTION DE NOMINA DE EMPLEADOS**

**INFORMACION DEL EMPLEADO**

D.N.I.	35653356	Buscar	C.U.I.L.	20356533567
NOMBRES	Jacinto			
APELLIDOS	Garcia			
DOMICILIO	Le Callecita			
FECHA DE NAC.	15-04-1990		FECHA DE ALTA	20-12-2023
OBRA ASIGNADA	Casa 1			
ART o SEGURO	Seguridad Total			
JORNAL (\$)	19000.0			

\* En el campo DNI y CUIL solo ingrese números sin ',' o '-'.

**FILTRAR POR OBRA**

ID	DNI	NOMBRES	APELLIDOS	OBRA ASIGNADA	JORNAL (\$)
1	33444555	Jose	Perez	Casa 1	18000.0
2	40700900	Pedro	Gonzalez	Casa 2	20000.0
3	35653356	Jacinto	Garcia	Casa 1	19000.0
4	39123321	Mario	Fernandez	Casa 2	21000.0
5	28001133	Martin	Gomez	Casa 3	23000.0
6	44959151	Maria Dolores	Snachez	Casa 3	22000.0

Puede modificar o dar de baja al registro.

#### OPCION 2:

**GESTION DE NOMINA DE EMPLEADOS**

**INFORMACION DEL EMPLEADO**

D.N.I.	1 40700900	Buscar	C.U.I.L.	23407009001
NOMBRES	Pedro			
APELLIDOS	Gonzalez			
DOMICILIO	Av. Perdida			
FECHA DE NAC.	02-08-1994		FECHA DE ALTA	07-12-2023
OBRA ASIGNADA	Casa 2			
ART o SEGURO	Seguridad Total			
JORNAL (\$)	20000.0			

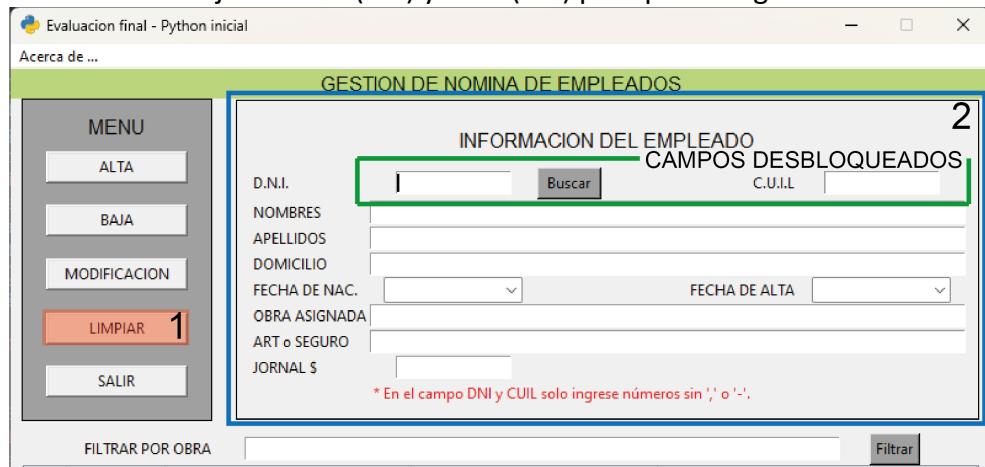
\* En el campo DNI y CUIL solo ingrese números sin ',' o '-'.

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

### 3.d. Limpieza de las cajas de entrada:

Luego de hacer una consulta de la información de algún registro se puede hacer el borrado de todos los campos (B) presionando en el botón de LIMPIEZA (A.4). Al vaciar los campos se volverán a habilitar las cajas de DNI (B.1) y CUIL (B.2) para poder ingresar datos.



### 3.e. Filtro por nombre de obra:

En el área del listado se propone la posibilidad de hacer un filtrado del mismo por el nombre de la obra asignada, y de esa manera tener la nómina correspondiente a cada obra.

Se debe completar el campo con el nombre de la obra (C.1) y luego presionar el botón de Filtrar (C.2).

Si se deja vacío el campo de filtrado y se presiona el botón Filtrar se volverá a mostrar todos los registros. En caso de errores será alertado con una ventana emergente y en todos los casos se le informará sobre el proceso en el área de notificación.



## CURSO PYTHON 3 – NIVEL INICIAL

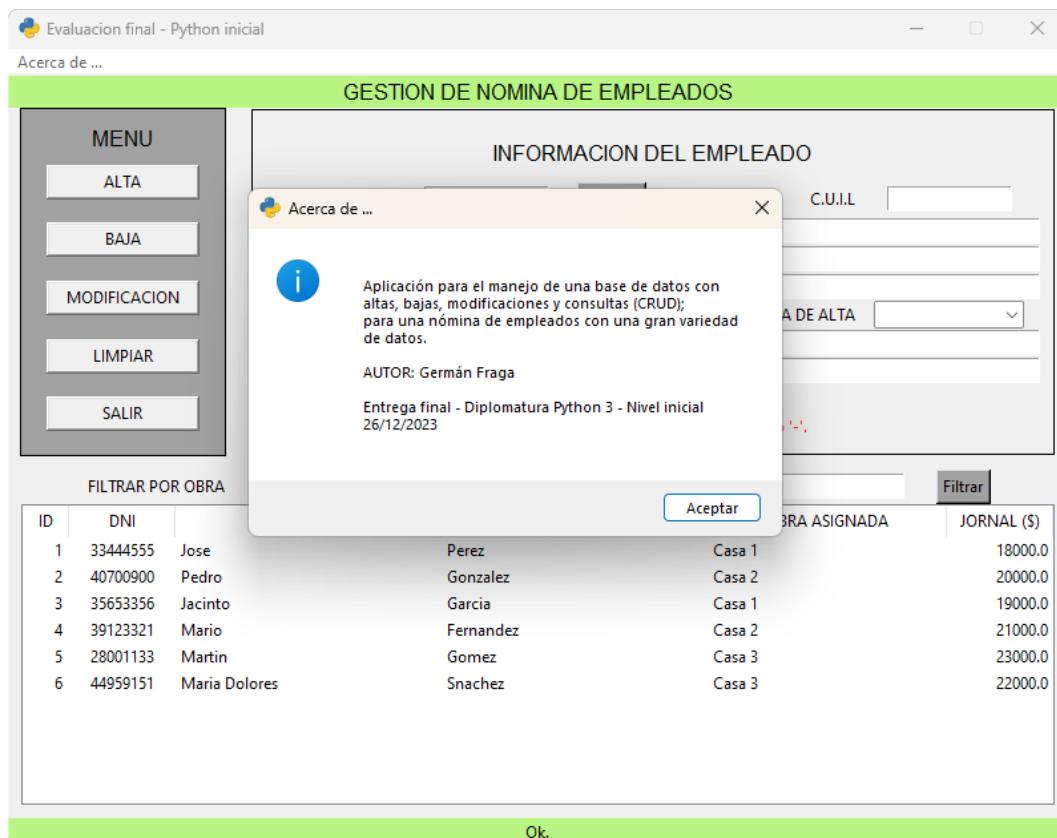
### ENTREGA FINAL

#### 3.f. Selección en listado:

Ya sea en el listado (C.3) completo o filtrado, al seleccionar un registro automáticamente se completarán todos los datos disponibles en el área de carga de datos. Para a posteriori poder ejecutar una BAJA o MODIFICACION.

#### 3.g Barra de menú.

Solo tiene la opción de “Acerca de ...” que hace abrir una ventana de información con una pequeña descripción de la aplicación y los datos del autor.



# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

### Detallado del código de la aplicación.

#### 4. Módulos de la aplicación

Se cargan los módulos de “tkinter” para el manejo de la interfaz gráfica, “Tkinter.messagebox” para las ventanas emergentes de alertas o decisiones, “Tkcalendar” para facilitar la carga de las fechas de nacimiento y alta, “os” para la obtención de las rutas dependiendo del sistema en el que se ejecute, “sqlite3” para el manejo de la base de datos y “re” para poder verificar la información con expresiones regulares.

```
3
4  from tkinter import *
5  from tkinter import ttk
6  from tkinter.messagebox import *
7  from tkcalendar import DateEntry
8  import sqlite3
9  import os
10 import re
11
```

#### 5. Detalle de las funciones del modelo

##### 5.a Funciones para manejo de la base de datos

###### *5.a.1 Creación de la base de datos.*

En primer lugar, se busca la ruta en la que se está ejecutando la aplicación y luego se concatena con la separación entre carpetas correspondiente al sistema sobre el cual se esta ejecutando y con el nombre de la carpeta donde se guardara la base de datos.

Se crea la base de datos o, si esta creada, se establece la conexión con la base dándole el nombre “nomina\_database.db”

```
12
13 # ***** FUNCIONES PARA MANEJO DE LA BASE DE DATOS *****
14 # ----- CONEXION CON LA BASE DE DATOS -----
15 def conect_database():
16     ruta = os.getcwd() + os.sep + "src" + os.sep
17     conexion = sqlite3.connect(ruta + "nomina_database.db")
18     return conexion
19
```

###### *5.a.2 Creación de la tabla.*

Se genera el cursor para apuntar a la base de datos, se asigna a la variable sql la instrucción de creación de la tabla, ejecuta el cursor y se cierra con el commit.

```
20
21 # ----- CREACION DE LA TABLA DE LA BASE DE DATOS -----
22 def create_table(conexion):
23     cursor = conexion.cursor()
24     sql = """CREATE TABLE empleados(
25             id INTEGER PRIMARY KEY AUTOINCREMENT,
26             dni INTEGER NOT NULL, cuil INTEGER NOT NULL,
27             nombres VARCHAR(30) NOT NULL, apellidos VARCHAR(30) NOT NULL,
28             domicilio VARCHAR(30), f_nacimiento VARCHAR(10), f_alta VARCHAR(10),
29             obra VARCHAR(30), art VARCHAR(30), jornal FLOAT)"""
30     cursor.execute(sql)
31     conexion.commit()
32
```

## CURSO PYTHON 3 – NIVEL INICIAL

### ENTREGA FINAL

#### 5.a.3 Consulta a la tabla de la base de datos.

Creo una función para las consultas a la base de datos, ya que es una tarea que se repite varias veces en el desarrollo de la aplicación. Recibe como parámetro un string con la instrucción de sqlite3 y retorna una lista con lo encontrado en la base se acuerda a los parámetros de búsqueda establecidos por cada función.

```
33 # —— CONSULTA A LA BASE DE DATOS ——
34 def update_table(sql: str):
35     conexion = conect_database()
36     cursor = conexion.cursor()
37     cursor.execute(sql)
38     data_list = cursor.fetchall()
39     return data_list
40
41
```

#### 5.a.4 Modificación de la tabla de la base de datos.

Es una función que es compartida entre el alta, la baja y las modificaciones. Por ello recibe un string con la instrucción de sqlite3 y una lista con los datos a grabar, modificar o eliminar en la tabla.

```
42 # —— MODIFICACION DE LA BASE DE DATOS ——
43 def modify_table(sql: str, data: list):
44     conexion = conect_database()
45     cursor = conexion.cursor()
46     cursor.execute(sql, data)
47     conexion.commit()
48
49
```

## 5.b Funciones de ABM

### 5.b.1 Alta de registro.

La información que recibe es una lista con todos los campos que se cargaron en el formulario de interfaz gráfica. Lo primero que se hace es comprobar que no estén vacíos los campos obligatorios. Luego se establecen los patrones de expresiones regulares para comprobar que el campo dni (B.1) solo tenga números entre 7 u 8 dígitos y que el campo de cuil (B.2) solo tenga números de 11 dígitos. Con esto correcto se establece la instrucción de sql que será enviada a la función de modificación de la tabla (5.a.4) junto con la lista de datos. Se vacían los campos del formulario (5.e.2), se informa el estado del proceso en la etiqueta al pie de la ventana y se actualiza el treeview (5.c.3). En el caso que no se carguen todos los datos obligatorios, se ingresen incorrectamente el dni y/o cuil o que el dni ya exista en la base de datos; se advierte al usuario y se especifica en la etiqueta al pie de la ventana.

```
50 # ***** FUNCIONES PARA ALTAS - BAJAS - MODIFICACIONES *****
51 # —— FUNCION ALTA DE REGISTRO ——
52 def create_record(data: list):
53     if not data[0] or not data[1] or not data[2] or not data[3]:
54         l_status.config(text="Complete todos los campos.", background="#FF5656")
55     else:
56         cadena_dni, cadena_cuil = data[0], data[1]
57         patron_dni, patron_cuil = "^\\d{7,8}$", "^\\d{11}$"
58         if (re.match(patron_dni, cadena_dni)) and (re.match(patron_cuil, cadena_cuil)):
59             sql = "SELECT * from empleados WHERE dni=" + data[0] + "';"
60             data_list = update_table(sql)
61
```

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
62         if not data_list:
63             sql = """INSERT INTO empleados(dni, cuil, nombres, apellidos,
64                                         domicilio, f_nacimiento, f_alta, obra, art, jornal)
65                                         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"""
66             modify_table(sql, data)
67             set_entry([["" for _ in range(11)] for _ in range(1)])
68             l_status.config(
69                 text="Los datos han sido guardados correctamente.",
70                 background="#B9F582",
71             )
72             update_treeview(tree)
73         else:
74             l_status.config(
75                 text="El DNI ingresado ya está cargado en la base de datos.",
76                 background="#FF5656",
77             )
78             showerror("ATENCIÓN !! ", "El DNI ingresado ya fue cargado.")
79     else:
80         l_status.config(
81             text="Verifique los datos ingresados.", background="#FF5656"
82         )
83         showerror("ATENCIÓN !! ", "La informacion cargada es incorrecta.")
84
```

### 5.b.2 Baja de registro.

Esta función recibe una lista con la información del registro a eliminar y el treeview. Se verifica que el campo de dni (B.1) no este vacío, se establece en la variable sql la instrucción de sqlite3 y se consulta al usuario si esta seguro de la eliminación. En caso afirmativo se envía a la función de modificación de la tabla (5.a.4) la instrucción de sql y el dato del dni en una tupla. A posteriori se actualiza el treeview (5.c.3), se vacían los campos del formulario (5.e.2) en la interfaz grafica y se notifica en la etiqueta al pie de la ventana del éxito de la operación. En el caso que el dni (B.1) este vacío o que se cancele la eliminación se alerta al usuario y se informa en la etiqueta al pie de la ventana.

```
86 # —— FUNCION DE BAJA DE REGISTRO ——
87 def delete_record(data: list, tree):
88     if not data[0]:
89         showerror("ATENCIÓN !! ", "No se ha seleccionado ningún registro.")
90         l_status.config(text="El campo DNI esta vacio.", background="#FF5656")
91     else:
92         sql = "DELETE FROM empleados WHERE dni = ?;"
93         option = askokcancel(
94             "Borra registro", "¿Está seguro que quiere eliminar ese registro?"
95         )
96         if option:
97             modify_table(sql, (data[0],))
98             update_treeview(tree)
99             set_entry([["" for _ in range(11)] for _ in range(1)])
100             l_status.config(
101                 text="Los datos han sido eliminados correctamente.",
102                 background="#B9F582",
103             )
104         else:
105             l_status.config(
106                 text="Se ha cancelado la eliminación de los datos.",
107                 background="#B9F582",
108             )
109
```

## CURSO PYTHON 3 – NIVEL INICIAL

### ENTREGA FINAL

#### 5.b.3 Modificación de un registro.

Al llamar a esta función se le envía la lista con los datos de los campos consultados y el treeview. Se verifica que el campo dni (B.1) no este vacío y de no estarlo se carga la instrucción de sqlite3 en la variable sql y se agrega el valor del dni al final de la lista data. Se consulta al usuario si esta de acuerdo con continuar el proceso y se de ser afirmativo a través de un bucle se eliminan los datos de dni y cui, que no se pueden modificar, de la lista; así de esta manera queda ordenada para enviar a la función de modificación de la tabla (5.a.4) junto con la instrucción de sql. Al regreso se vacían los campos del formulario (5.e.2), se actualiza el treeview (5.c.3) y se notifica al usuario, en la etiqueta al pie de la ventana, que la operación se concretó. Si el campo dni (B.1) está vacío o se cancela la modificación se alerta al usuario y se informa en la etiqueta al pie de la ventana.

```
110
111 # —— FUNCION DE MODIFICACION DE REGISTROS ——
112 def modify_record(data: list, tree):
113     if not data[0]:
114         showerror("ATENCIÓN!! ", "No se ha seleccionado ningún registro.")
115         l_status.config(text="El campo DNI esta vacio.", background="#FF5656")
116     else:
117         sql = """UPDATE empleados SET nombres = ?, apellidos = ?, domicilio = ?,
118                                     f_nacimiento = ?, f_alta = ?, obra = ?, art = ?,
119                                     jornal = ? WHERE dni = ?;"""
120         data.append(data[0])
121         option = askokcancel(
122             "Modifica registro", "¿Está seguro que quiere modificar ese registro?"
123         )
124         if option:
125             for _ in range(2):
126                 data.pop(0)
127             modify_table(sql, data)
128             set_entry([[ "" for _ in range(11)] for _ in range(1)])
129             l_status.config(
130                 text="Los datos han sido modificados correctamente.",
131                 background="#B9F582",
132             )
133             update_treeview(tree)
134         else:
135             l_status.config(
136                 text="Se ha cancelado la modificación de los datos.",
137                 background="#B9F582",
138             )
139
```

#### 5.c. Funciones de consulta y treeview.

##### 5.c.1 Consulta a través del treeview.

Al seleccionar una fila del listado (C.3) se toma el dato del id del treeview y se le establece a la variable sql un string con la instrucción de sqlite3 para consultar ese id a la base de datos. Para ello se envia la variable sql y el id a la función de consulta de la base (5.a.3) retornándonos una lista con toda la información para enviar a la función de seteo de los campos del formulario (5.e.2).

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
140 # ***** FUNCIONES PARA CONSULTAS Y TREEVIEW *****
141 # ----- FUNCION DE CONSULTA DESDE TREEVIEW -----
142 def consult_record(event):
143     item = tree.item(tree.selection())
144     data = int(item["text"])
145     sql = "SELECT * FROM empleados WHERE id = " + str(data)
146     data_list = update_table(sql)
147     set_entry(data_list)
148
149
```

### 5.c.2 Búsqueda desde el botón del formulario.

Esta función recibe un string que funcionara como índice para poder hacer la búsqueda de la información en la base de datos. Para ello se establece en sql la instrucción de sqlite3 para luego enviarla junto con el índice a la función de consulta de la base (5.a.3), recibiendo una lista con todos los campos a cargar a través de la función de seteo de los campos (5.e.2). En caso que no se reciba información se le alerta al usuario que no está registrado ese dni. En ambos casos se le informa de lo acontecido en la etiqueta al pie de la ventana.

```
150 # ----- FUNCION DE BUSQUEDA -----
151 def search_record(indice: str):
152     sql = "SELECT * from empleados WHERE dni='{}'".format(indice)
153     data_list = update_table(sql)
154     if not data_list:
155         l_status.config(
156             text="No se encontró el DNI solicitado en la base de datos.",
157             background="#ff1b1b",
158         )
159         showerror("ATENCIÓN!! ", "Este DNI no existe en la base de datos")
160     else:
161         l_status.config(
162             text="La búsqueda se concreto correctamente.", background="#B9F582"
163         )
164     set_entry(data_list)
165
166
```

### 5.c.3 Actualización de la información en el treeview.

Esta función puede ser llamada desde otras funciones o desde el botón de filtrar por obra, por eso recibe el tree y un parámetro que establece desde donde surge la consulta y así poder establecer correctamente la instrucción de sql en la variable con el mismo nombre. Se envía la instrucción a la función de consulta de la base de datos (5.a.3), retornándonos una lista con todos los campos que se guarda en data\_list. Se borran todos los datos que hay en el treeview a través de un bucle for y con la misma metodología se cargan los datos recuperados de la base de datos. Para finalizar de vacía la entrada del filtro.

```
167 # ----- FUNCION ACTUALIZAR TREEVIEW -----
168 def update_treeview(mitreview, parameter=None):
169     if not parameter:
170         sql = "SELECT id, dni, nombres, apellidos, obra, jornal FROM empleados"
171     else:
172         sql = (
173             "SELECT id, dni, nombres, apellidos, obra as 'Obra', jornal FROM empleados WHERE obra='"
174             + parameter
175             + "'";
176         )
177
```

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
178     data_list = update_table(sql)
179
180     records = mitreview.get_children()
181     for element in records:
182         mitreview.delete(element)
183
184     for row in data_list:
185         mitreview.insert(
186             "", 0, text=row[0], values=(row[1], row[2], row[3], row[4], row[5]))
187
188     var_filtro.set("")
189
```

### 5.d Función de cierre de la aplicación.

Al presionar el botón de salida (A.5) se dispara esta función que consulta al usuario si está seguro de continuar, y de ser así se cierra la aplicación.

```
190
191 # —— FUNCION DE CIERRE DE LA APLICACION ——
192 def close_app():
193     option = askokcancel("Cerrar la aplicación", "¿Está seguro que quiere salir?")
194     if option:
195         window.destroy()
196
```

### 5.e Funciones de manipulación de datos.

#### 5.e.1 Crear lista.

Diseñe esta función para crear una lista con todos los campos del formulario (B) de la interfaz gráfica y sea más sencilla la manipulación de los datos entre función y función. En el dato de la obra asignada se le aplica un método “.capitalize” para estandarizar el dato y que sea más efectiva la coincidencia a la hora de efectuar una búsqueda.

```
197
198 # ***** MANIPULACION DE DATOS *****
199 # —— CREACION DE UNA LISTA PARA MOVIMIENTO DE LOS DATOS ——
200 def create_list():
201     data_list = [
202         var_dni.get(),
203         var_cuil.get(),
204         var_nombre.get(),
205         var_apellido.get(),
206         var_domicilio.get(),
207         var_fnacimiento.get(),
208         var_falta.get(),
209         var_obra.get().capitalize(),
210         var_art.get(),
211         var_jornal.get(),
212     ]
213     return data_list
214
```

#### 5.e.2 Seteo de los entrys del formulario.

Creo esta función debido a que esta tarea se usa en varias funciones y evita tener que repetir estas líneas varias veces en el código. En ella se puede cargar datos o vaciar los entrys. A su vez dependiendo si está vacío o con información el campo dni lo habilita o deshabilita respectivamente.

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
216 # —— SETEO DE LOS ENTRY ——
217 def set_entry(data_list: list):
218     var_dni.set(data_list[0][1])
219     var_cuil.set(data_list[0][2])
220     var_nombre.set(data_list[0][3])
221     var_apellido.set(data_list[0][4])
222     var_domicilio.set(data_list[0][5])
223     var_fnacimiento.set(data_list[0][6])
224     var_falta.set(data_list[0][7])
225     var_obra.set(data_list[0][8])
226     var_art.set(data_list[0][9])
227     var_jornal.set(data_list[0][10])
228     if not data_list[0][1]:
229         l_status.config(text="Ok.", background="#B9F582")
230         e_dni.config(state="normal")
231         e_cuil.config(state="normal")
232     else:
233         l_status.config(
234             text="Puede modificar o dar de baja al registro.", background="#B9F582"
235         )
236         e_dni.config(state="disabled")
237         e_cuil.config(state="disabled")
238
```

### 5.f Declaración de la variable info.

Se le da un texto multilinea para mostrar en la ventana que se abre a través del menubar de “Acerca de ...”

```
239
240 # —— DECLARACION DE TEXTO PARA VENTANA ACERCA DE ... ——
241 info = """
242     Aplicación para el manejo de una base de datos con
243     altas, bajas, modificaciones y consultas (CRUD);
244     para una nómina de empleados con una gran variedad
245     de datos.
246
247     AUTOR: Germán Fraga
248
249     Entrega final - Diplomatura Python 3 - Nivel inicial
250     26/12/2023
251 """
252
```

## 6. Vista y control

### 6.a Raíz de la ventana.

Se declara la ventana principal de la aplicación determinando el título, el icono y la configuración que restringe el cambio de las dimensiones de la misma.

Para la búsqueda del ícono en la carpeta “img”, uso el módulo os para establecer la ruta y el símbolo de separación de carpetas que corresponde al sistema operativo sobre el que se corre la aplicación.

Se dibuja el menú con solo la opción de “Acerca de ...” que dispara una ventana de información con el texto antes declarado (5.f).

Se determina la etiqueta para el título de la aplicación y se dibuja a través del método de grid.

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
253 # ***** VISTA Y CONTROL *****
254
255 window = Tk()
256 window.title("Evaluacion final - Python inicial")
257 # window.geometry("810x573")
258 window.resizable(False, False)
259 ruta = os.getcwd() + os.sep + "img" + os.sep
260 window.iconbitmap(ruta + "python.ico")
261
262 menubar = Menu(window, relief="solid")
263 menubar.add_cascade(
264     label="Acerca de ... ", command=lambda: showinfo("Acerca de ... ", info)
265 )
266 window.config(menu=menubar)
267
268 Label(window, text="GESTION DE NOMINA DE EMPLEADOS", bg="#B9F582", font="Bold").grid(
269     row=0, column=0, columnspan=2, sticky=W + E
270 )
271
```

Declaro todas las variables que serán utilizadas en la interfaz gráfica.

```
271
272 # ----- DEFINICIONN DE VARIABLES -----
273 var_dni, var_cuil = StringVar(), StringVar()
274 var_nombre, var_apellido, var_domicilio = StringVar(), StringVar(), StringVar()
275 var_fnacimiento, var_falta = StringVar(), StringVar()
276 var_obra, var_art = StringVar(), StringVar()
277 var_jornal, var_filtro = StringVar(), StringVar()
278
```

La ventana principal se va a dividir en tres frames que contendrán las distintas áreas de la pantalla.

### 6.b Frame de menú. (A)

Este va a contener los botones para el alta (A.1), baja (A.2), modificación (A.3), limpieza (A.4) y salida (A.5) de la aplicación. Cada uno llamará a la función correspondiente a través de una función lambda que permite el envío de la información. Todos los botones son dibujados en el frame con el método de grid.

```
278
279 # ----- FRAME DE MENU -----
280 frame_menu = Frame(window, bg="#c8c8c8", padx=10, pady=10, bd=1, relief="solid")
281 frame_menu.grid(row=1, column=0)
282
283 Label(frame_menu, text="MENU", bg="#c8c8c8", font="Bold").grid(
284     row=0, column=0, sticky=W + E
285 )
286
287 btn_alta = Button(
288     frame_menu, text="ALTA", width=15, command=lambda: create_record(create_list()))
289
290 btn_alta.grid(row=1, column=0, padx=9, pady=8)
291 btn_baja = Button(
292     frame_menu,
293     text="BAJA",
294     width=15,
295     command=lambda: delete_record(create_list(), tree),
296 )
```

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
297 btn_baja.grid(row=2, column=0, padx=2, pady=9)
298 btn_modificacion = Button(
299     frame_menu,
300     text="MODIFICACION",
301     width=15,
302     command=lambda: modify_record(create_list(), tree),
303 )
304 btn_modificacion.grid(row=3, column=0, padx=2, pady=9)
305 btn_consulta = Button(
306     frame_menu,
307     text="LIMPIAR",
308     width=15,
309     command=lambda: set_entry([[ "" for _ in range(11)] for _ in range(1)]),
310 )
311 btn_consulta.grid(row=4, column=0, padx=2, pady=9)
312 btn_cerrar = Button(frame_menu, text="SALIR", width=15, command=lambda: close_app())
313 btn_cerrar.grid(row=5, column=0, padx=2, pady=8)
314
```

### 6.c Frame de datos. (B)

Aquí se dibujan con el método de grid todas las cajas de entrada de datos del formulario con sus respectivas etiquetas y el botón para poder disparar la función de búsqueda (B.10) del dni cargado.

```
314 # ----- FRAME DE DATOS -----
315 frame_datos = Frame(window, padx=10, pady=10, bd=1, relief="solid")
316 frame_datos.grid(row=1, column=1)
317
318 Label(frame_datos, text="INFORMACION DEL EMPLEADO", font="Bold").grid(
319     row=0, column=0, columnspan=6, pady=10, sticky=W + E
320 )
321
322 Label(frame_datos, text="D.N.I.").grid(row=1, column=0, sticky=W)
323 Label(frame_datos, text="C.U.I.L").grid(row=1, column=4, sticky=E)
324 Label(frame_datos, text="NOMBRES").grid(row=2, column=0, sticky=W)
325 Label(frame_datos, text="APELLIDOS").grid(row=3, column=0, sticky=W)
326 Label(frame_datos, text="DOMICILIO").grid(row=4, column=0, sticky=W)
327 Label(frame_datos, text="FECHA DE NAC.").grid(row=5, column=0, sticky=W)
328 Label(frame_datos, text="FECHA DE ALTA").grid(row=5, column=4, sticky=E)
329 Label(frame_datos, text="OBRA ASIGNADA").grid(row=6, column=0, sticky=W)
330 Label(frame_datos, text="ART o SEGURO").grid(row=7, column=0, sticky=W)
331 Label(frame_datos, text="JORNAL $").grid(row=8, column=0, sticky=W)
332 Label(
333     frame_datos,
334     text="* En el campo DNI y CUIL solo ingrese números sin ',' o '-' .",
335     fg="#ff1b1b",
336 ).grid(row=9, column=1, columnspan=6, sticky=W)
337
338 e_dni = Entry(frame_datos, textvariable=var_dni, width=15)
339 e_dni.grid(row=1, column=1)
340 e_cuil = Entry(frame_datos, textvariable=var_cuil, width=15)
341 e_cuil.grid(row=1, column=5)
342 e_nombre = Entry(frame_datos, textvariable=var_nombre, width=80)
343 e_nombre.grid(row=2, column=1, columnspan=5)
344 e_apellido = Entry(frame_datos, textvariable=var_apellido, width=80)
345 e_apellido.grid(row=3, column=1, columnspan=5)
346 e_direccion = Entry(frame_datos, textvariable=var_domicilio, width=80)
347 e_direccion.grid(row=4, column=1, columnspan=5)
348 e_fnacimiento = DateEntry(
349     frame_datos,
350     width=15,
```

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
352     justify="center",
353     date_pattern="dd-mm-yyyy",
354     textvariable=var_fnacimiento,
355     foreground="#000000",
356 )
357 e_fnacimiento.grid(row=5, column=1)
358 e_falta = DateEntry(
359     frame_datos,
360     width=15,
361     justify="center",
362     date_pattern="dd-mm-yyyy",
363     textvariable=var_falta,
364     foreground="#000000",
365 )
366 e_falta.grid(row=5, column=5)
367 e_obra = Entry(frame_datos, textvariable=var_obra, width=80)
368 e_obra.grid(row=6, column=1, columnspan=5)
369 e_art = Entry(frame_datos, textvariable=var_art, width=80)
370 e_art.grid(row=7, column=1, columnspan=5)
371 e_jornal = Entry(frame_datos, textvariable=var_jornal, width=15)
372 e_jornal.grid(row=8, column=1)
373
374 btn_buscar = Button(
375     frame_datos, text="Buscar", width=6, command=lambda: search_record(var_dni.get())
376 )
377 btn_buscar.grid(row=1, column=2, sticky=W)
378
```

### 6.d Frame treeview. (C)

Por último, en este contendrá el listado (C.3) con toda la información de la base de datos y una caja de entrada de datos (C.1) para poder hacer un filtrado del listado por el nombre de la obra asignada. Al cual le corresponde un botón (C.2) que dispara la actualización del treeview (5.c.3) filtrando la búsqueda por el dato ingresado, y si este está vacío se cancela el filtrado. Al dato cargado se le aplica un metodo “.capitalize” para que haya mejores correspondencias con los campos de la base de datos. Todos estos widgets se ubican con el método de grid.

```
378 # ----- FRAME PARA TREEVIEW -----
379 frame_tree = Frame(window, padx=10, pady=10, bd=0, relief="solid")
380 frame_tree.grid(row=2, column=0, columnspan=2)
381 frame_tree.config(width=800, height=180)
382
383 Label(frame_tree, text="FILTRAR POR OBRA").grid(row=0, column=0, sticky=E)
384
385 e_filtro = Entry(frame_tree, textvariable=var_filtro, width=80)
386 e_filtro.grid(row=0, column=1)
387
388 btn_filtrar = Button(
389     frame_tree,
390     text="Filtrar",
391     command=lambda: update_treeview(tree, var_filtro.get().capitalize()),
392 )
393 btn_filtrar.grid(row=0, column=2, sticky=W)
394
395 tree = ttk.Treeview(frame_tree)
396 tree["columns"] = ("col1", "col2", "col3", "col4", "col5")
397 tree.column("#0", width=35, minwidth=20, anchor="center")
398 tree.column("col1", width=80, minwidth=50, anchor="center")
399 tree.column("col2", width=200, minwidth=100)
400 tree.column("col3", width=200, minwidth=100)
401 tree.column("col4", width=180, minwidth=100)
402 tree.column("col5", width=80, minwidth=60, anchor=E)
```

# CURSO PYTHON 3 – NIVEL INICIAL

## ENTREGA FINAL

```
404     tree.heading("#0", text="ID")
405     tree.heading("col1", text="DNI")
406     tree.heading("col2", text="NOMBRES")
407     tree.heading("col3", text="APELLOS")
408     tree.heading("col4", text="OBRA ASIGNADA")
409     tree.heading("col5", text="JORNAL ($)")
410     tree.grid(row=1, column=0, columnspan=3)
411     tree.bind("<ButtonRelease-1>", consult_record)
412
```

### 6.e Label de status. (D)

Se inicializa la etiqueta que luego se utilizara continuamente para informar al usuario de los que va pasando en el proceso de utilización de la aplicación.

```
412
413 # —— LABEL DE STATUS ——
414 l_status = Label(window, text="Ok.", bg="#B9F582")
415 l_status.grid(row=3, column=0, columnspan=2, sticky=W + E)
416
```

Se intenta (try:) conectar a la base de datos y crear la tabla, llamando a las funciones correspondientes en cada caso. En el caso de no poder llevar a cabo estas sentencias se hace una excepción (except:) para controlar el error. En ambos casos se configura con la leyenda correspondiente la etiqueta de status al pie de la aplicación.

```
416
417 try:
418     conexion = conect_database()
419     create_table(conexion)
420     l_status.config(text="Se ha creado correctamente la base de datos.")
421 except:
422     l_status.config(
423         text="La base de datos ya está creada. Se ha accedido correctamente."
424     )
425
426 update_treeview(tree)
427
428 window.mainloop()
```

Por ultimo y muy importante se efectúa el mainloop para poder mantener la ventana activa hasta salir de la aplicación.