

**PRODUCTO INTEGRADOR DE APRENDIZAJE**

Nombre de la Unidad de Aprendizaje: **\_\_\_**Programación avanzada**\_\_\_**

Nombre del proyecto: **\_\_**Análisis de caso de aplicación: Administrador\_\_\_\_\_

Programa educativo: **\_\_**Licenciado en Tecnologías de Información\_\_\_\_\_\_\_\_\_

Semestre: **\_\_**2do\_\_ Grupo: **\_\_\_**22**\_\_\_**

Nombre del maestro: **\_\_\_\_**Hiram Isaí Pérez Sauceda**\_\_\_\_\_**

Nombre de los integrantes del equipo:

(ESCAMILLA TAPIA FERNANDA YAMILETH- # 1904359)

(URIBE SALAZAR ANTONIO DE JESUS- # 1805019)

|  |  |
| --- | --- |
| **Contenido mínimo a evaluar** | **Cumplimiento** |
| Índice |  |
| Introducción. - incluye valores UANL aplicados |  |
| Análisis y emisión de juicio |  |
| Conclusiones individuales |  |
| Conclusión del equipo |  |
| Actividad en inglés |  |
| Identificación de sub resultados de aprendizaje ANECA. |  |
| Calificación PIA: |  |
| Firma del maestro | |

San Nicolás de los Garza, ciudad universitaria a (20-noviembre-2020)

**“ÍNDICE”**

**INTRODUCCIÓN**……………………………………………………………………..….03

**ANÁLISIS DE CASO DE APLICACIÓN: ADMINISTRACIÓN Y GESTIÓN EN UNA EMPRESA**………………………………………………………………………….04

*ANECA 2.4 ……………………………………………………………………………….04*

*MANUAL DE USUARIO…………………………………………………………………04*

*ALGORITMOS…………………………………………………………………………...13*

*ANECA 4.1 ……………………………………………………………………………….33*

**CONCLUSIONES INDIVIDUALES**…………………………………………………….*34*

**CONCLUSIÓN** (español/Inglés)………………………………………………………..*35*

**“INTRODUCCIÓN”**

En la actualidad los beneficios que ofrece la tecnología no se limitan solo a un campo en específico, si no que ésta ayuda en todos los aspectos posibles, tanto en educación, entretenimiento, como en la vida diaria. Sin embargo, no debemos de olvidar que la programación ha sido la causante de que la tecnología haya podido avanzar hasta como la conocemos en la actualidad.

Las soluciones que brinda el desarrollo de nuevo sistemas son aplicadas en diversas áreas, tal vez no lo notamos, pero en todos los campos de trabajo se necesita tener un sistema, un ejemplo sería en la medicina, cada hospital necesita tener un sistema que le ayude a llevar un control sobre cada paciente, medico, medicina, y todos los aspectos que sean considerados como importantes, la tarea de crear un sistema optimo y eficiente la deben de realizar los desarrolladores.

A lo largo de este trabajo, expondremos el software que hemos desarrollado, en el cual aplicamos todos los conocimientos adquiridos a lo largo del semestre, nuestro algoritmo esta orientado a la aplicación de un campo de gestión empresarial.

Es importante recalcar que para realizar esta actividad tuvimos que poner en práctica algunos valores enseñados por nuestra *alma máter,* los valores que tomamos en cuenta para realizar nuestro proyecto son: *honestidad*: al desarrollar nuestro propio trabajo desde cero, sin ayuda de terceras personas o sin plagiar algún trabajo ya realizado; *solidaridad*: debido a que ambos integrantes del equipo trabajamos en conjunto para alcanzar un mismo objetivo; por último, *justicia:* debido a que delegamos las actividades de una manera que la dificultas fuera proporcional.

**“ANÁLISIS DE CASO DE APLICACIÓN: *ADMINISTRACIÓN Y GESTIÓN EN UNA EMPRESA*”**

La funcionalidad de Python es crear y manipular el sistema en manera formal para otros usuarios. Creando una base de arquitectura de ventanas, librerías, fuentes etc.

Tiene la complejidad de desarrollar este software es las decisiones del cliente o la necesidad a conocer; es por ellos implementar toda la característica que debe completar para cualquier necesidad o error presentado.

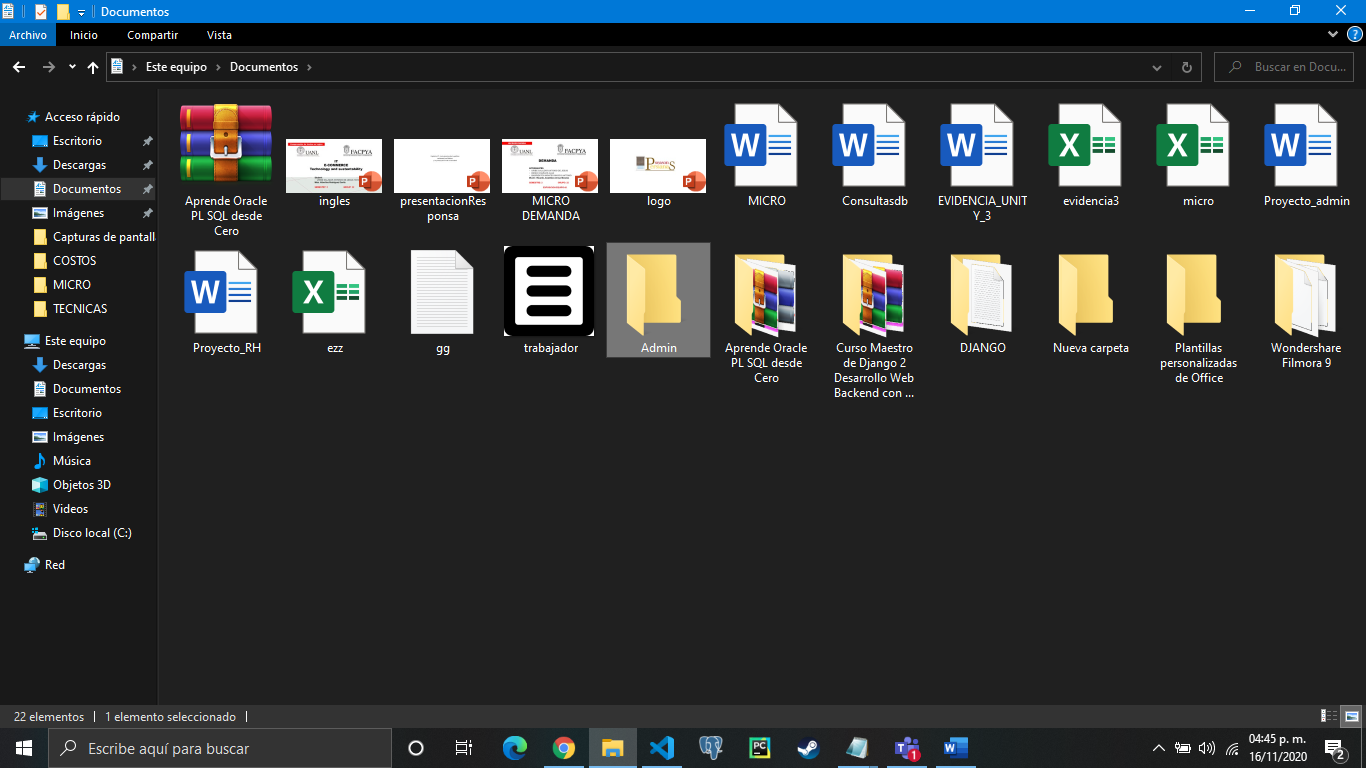
El uso de POO ayudo a formar varias bases para construir los botones y funciones de cada pestaña o ventana.

**ANECA 2.4 ESCOGER LOS PATRONES DE SOLUCIÓN, ALGORITMOS Y ESTRUCTURAS DE DATOS APROPIADOS.**

El programa presentado esta diseñado con el objetivo de ser utilizado en un campo de gestión empresarial, más específicamente para los usuarios del área de recursos humanos. Mediante este programa el área de recursos humanos podrá tener un control y por lo tanto gestionar tantos empleados, departamentos, jefes y tener un control sobre el personal de la empresa. Tener un sistema bien estructurado es fundamental para todas las empresas debido a que mediante el pueden dar de alta, de baja, consultar datos, o modificar datos de sus trabajadores.

Todo este recurso es utilizado en la elaboración del algoritmo y formar un orden para el usuario requerido o manipulador del sistema.

A continuación, prestaremos el manual especial del programa; para que cada usuario conozca los pasos para manipular dicho programa.

**Manual de usuario.**

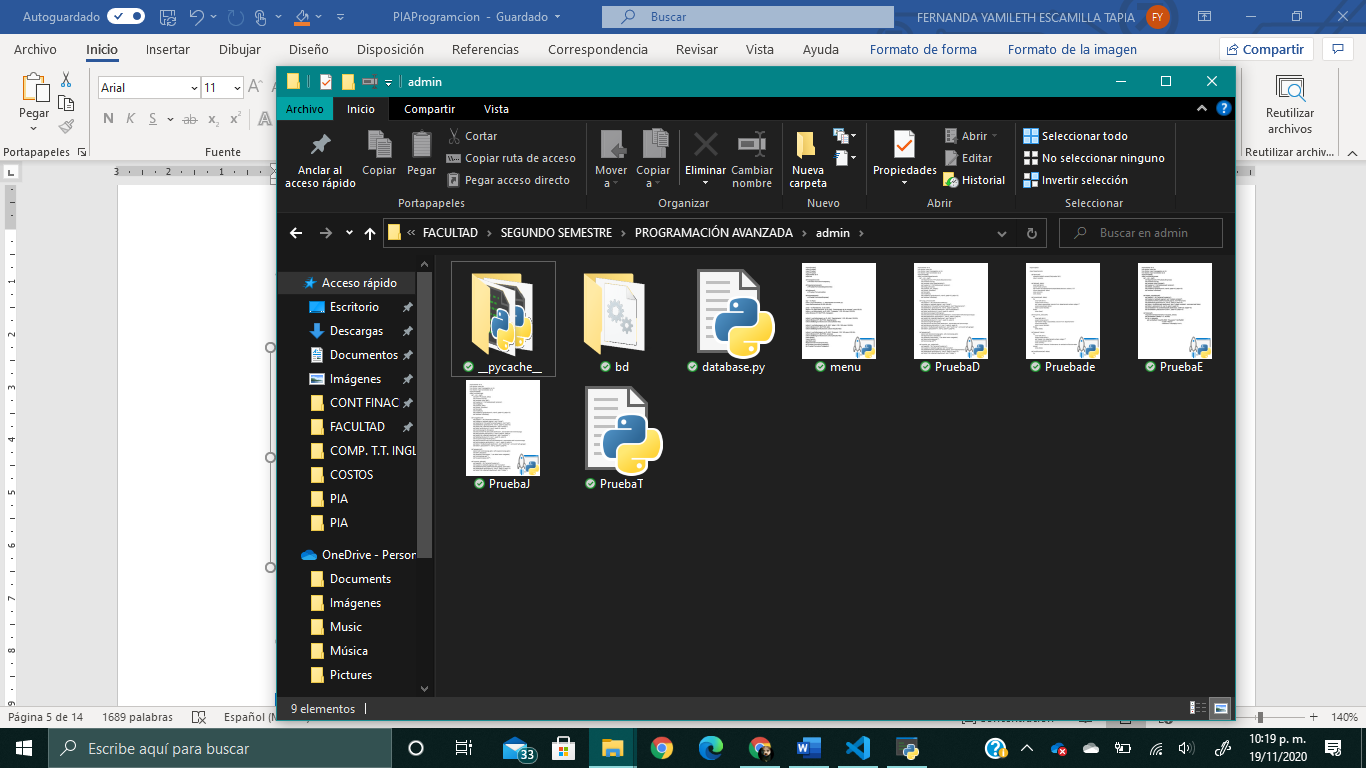
**Pasos**

*1-Primer los archivos en una ubicación personal o publica para ser ejecutado.*

En nuestro caso lo guardamos en el apartado de documentos con un nombre especial (admin).

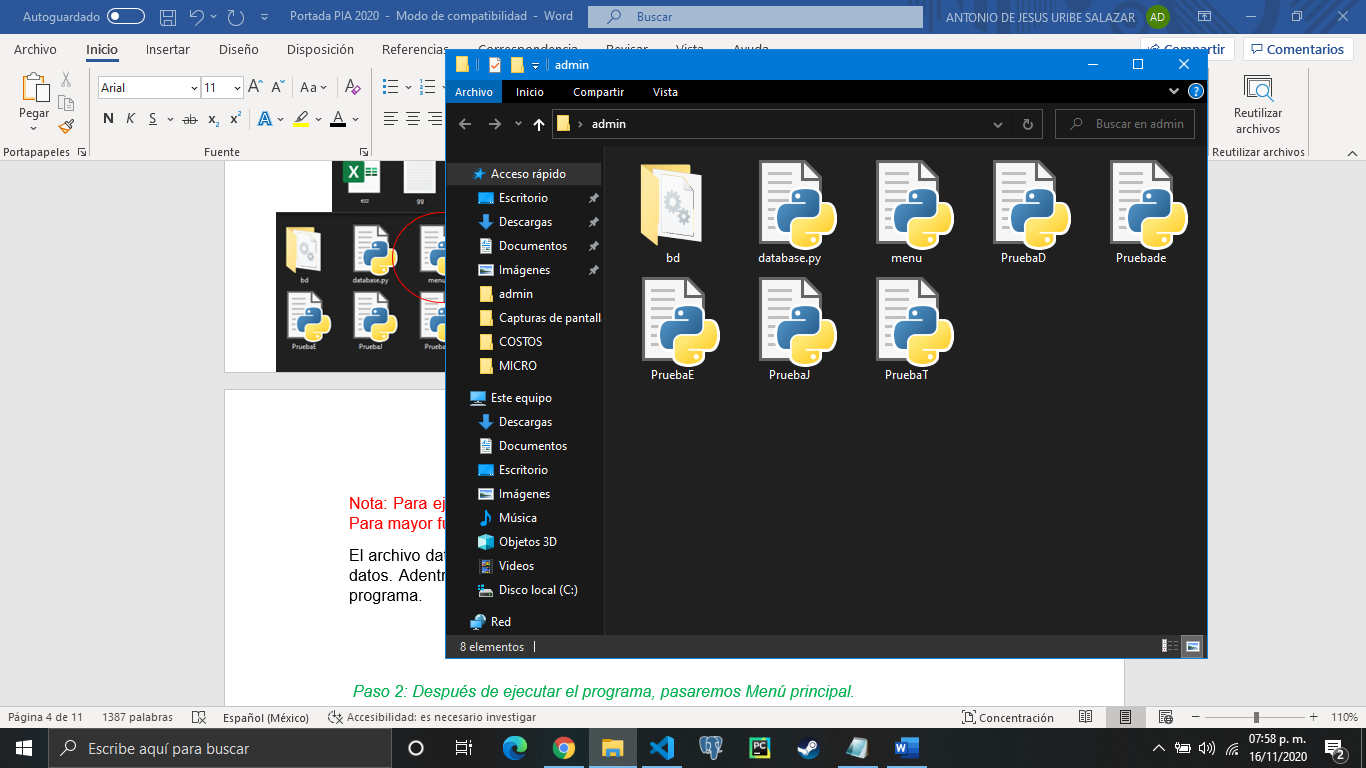
Después de hacer una carpeta, dentro de ella podemos encontrar una cantidad de archivos. El archivo principal que ejecutara nuestro programa se llama menú.

Es ahí donde cualquier compilador Python podrás ejecutar.



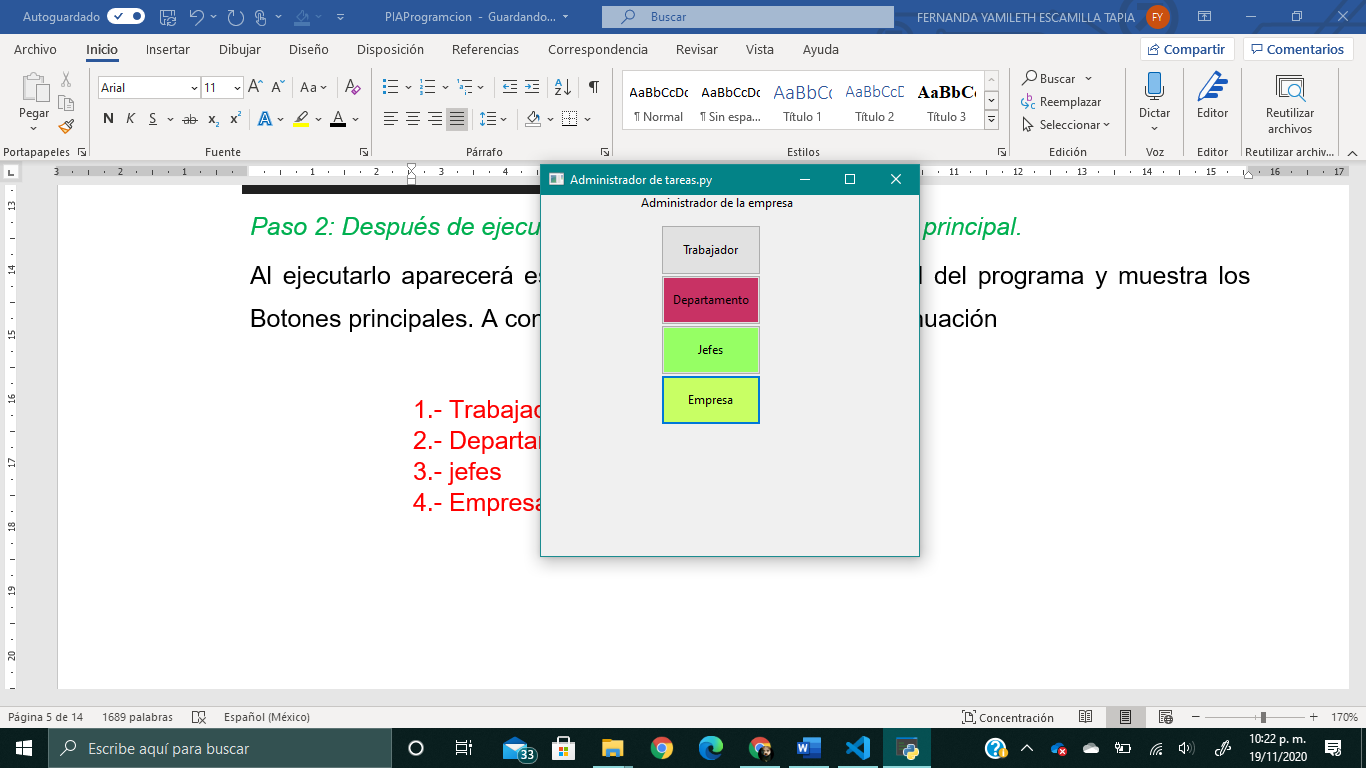
*Nota: Para ejecutarlo de manera correcta tendrás que descargar la librería de Wxpython. Para mayor función.*

El archivo databases.py es el archivo principal para el soporte técnico de crear la base de datos. Adentro del programa contiene las indicaciones necesarias para seguir creando el programa.



*Paso 2: Después de ejecutar el programa, pasaremos Menú principal.*

Al ejecutarlo aparecerá esta ventana. Es el menú principal del programa y muestra los Botones principales. A continuación, lo mostraremos a continuación



1.- Trabajador

2.- Departamento

3.- jefes

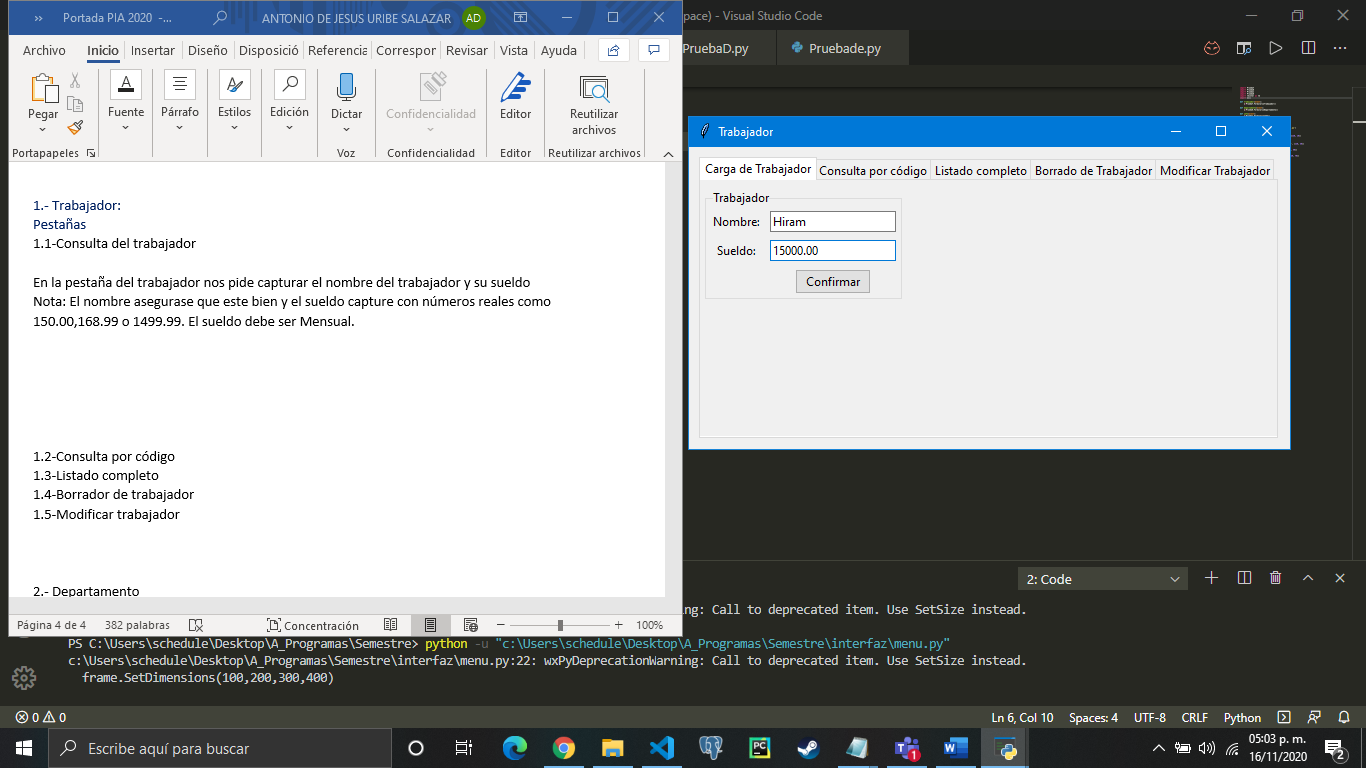
4.- Empresa

**1.- Trabajador:**

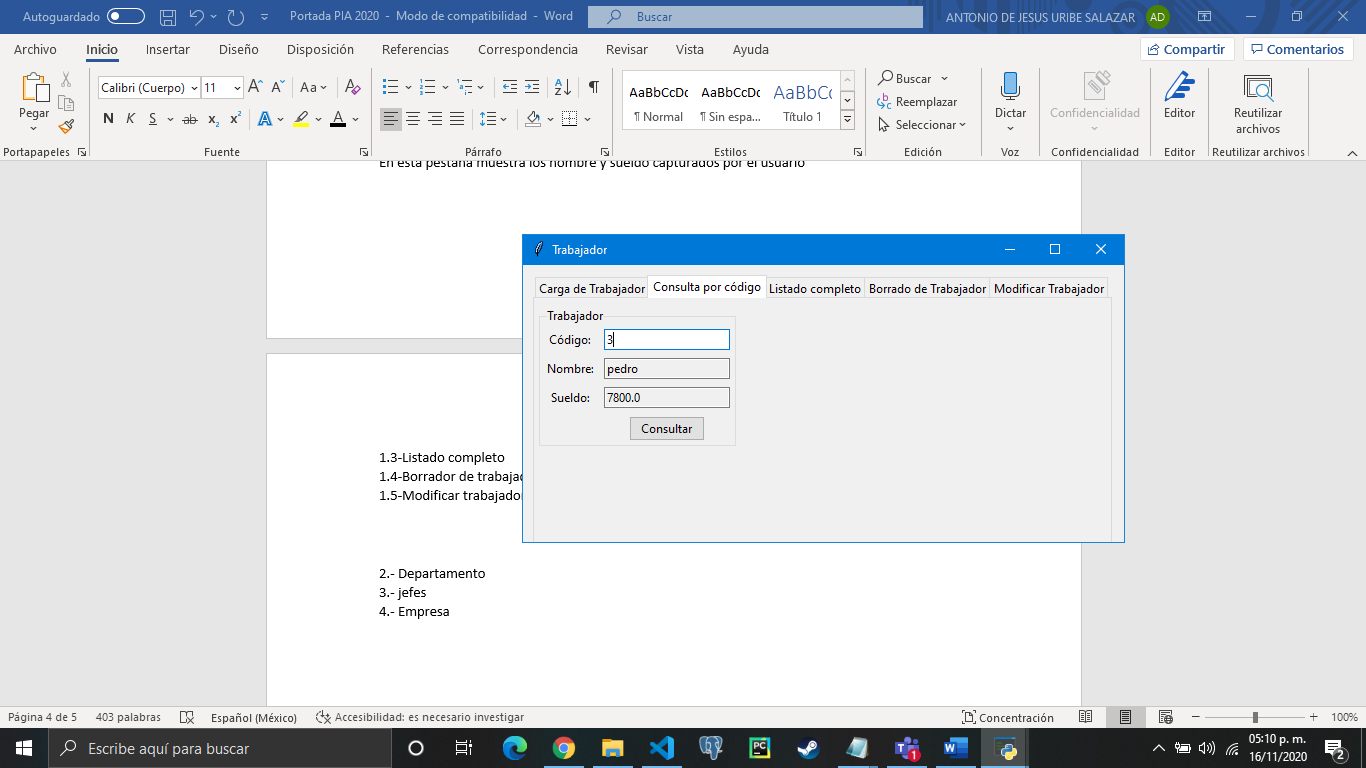
Pestañas

**1.1-Consulta del trabajador**

En la pestaña del trabajador nos pide capturar el nombre del trabajador y su sueldo

Nota: El nombre asegurase que este bien y el sueldo capture con números reales como 150.00,168.99 o 1499.99. El sueldo debe ser Mensual.

Después para continuar, daremos clic en Confirmar para continuar.

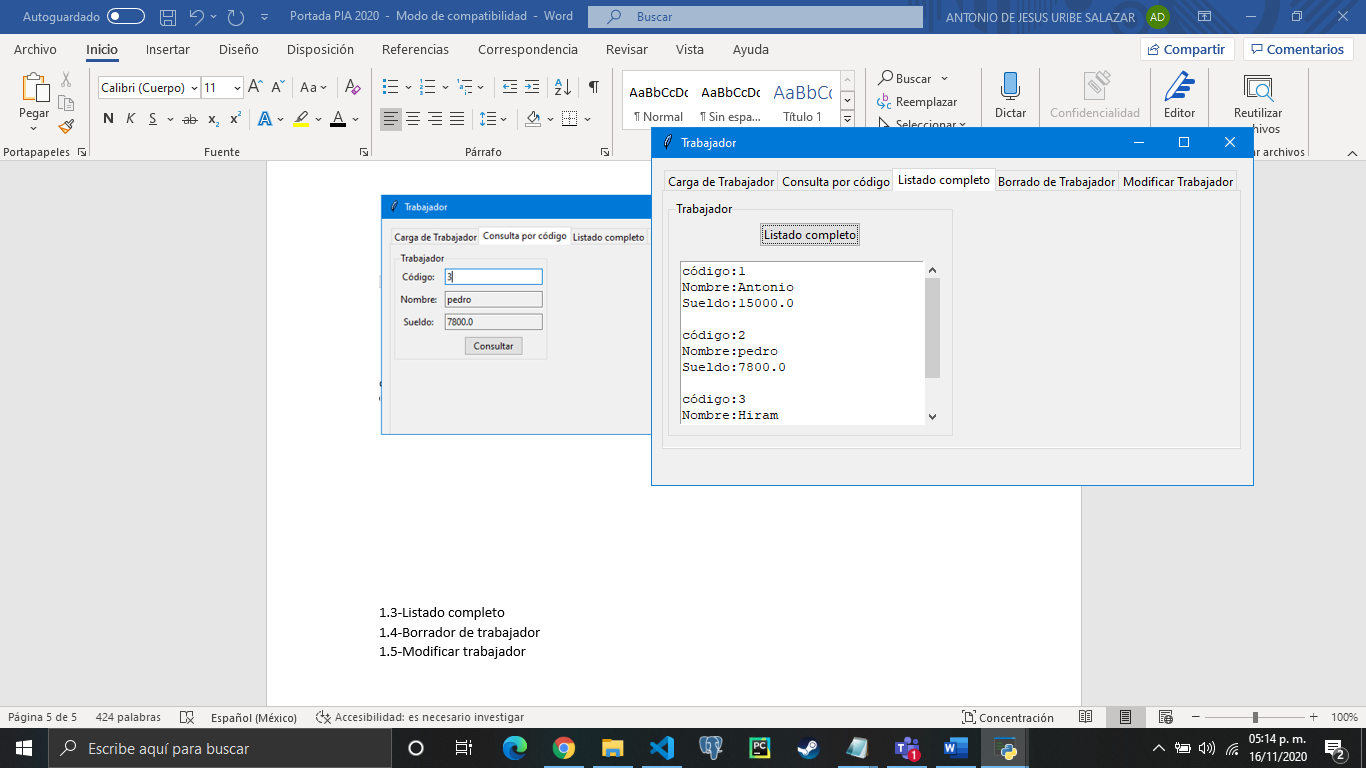
**1.2-Consulta por código**

En esta pestaña muestra los nombre y sueldo capturados por el usuario

Al insertar el código grabado por el sistema y dar clic en Consultar, aparecerá en Nombre y sueldo los campos correspondientes.

**1.3-Listado completo**

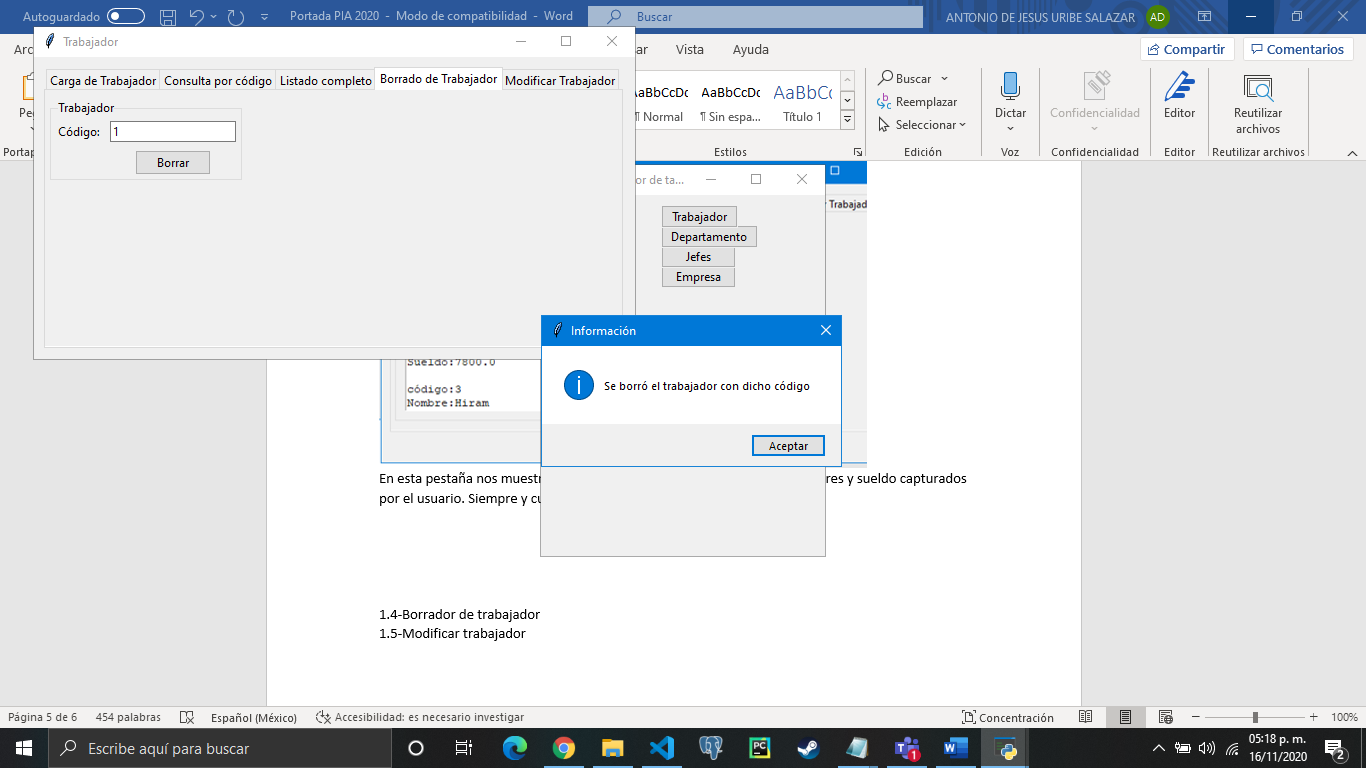
Figura listado completo.



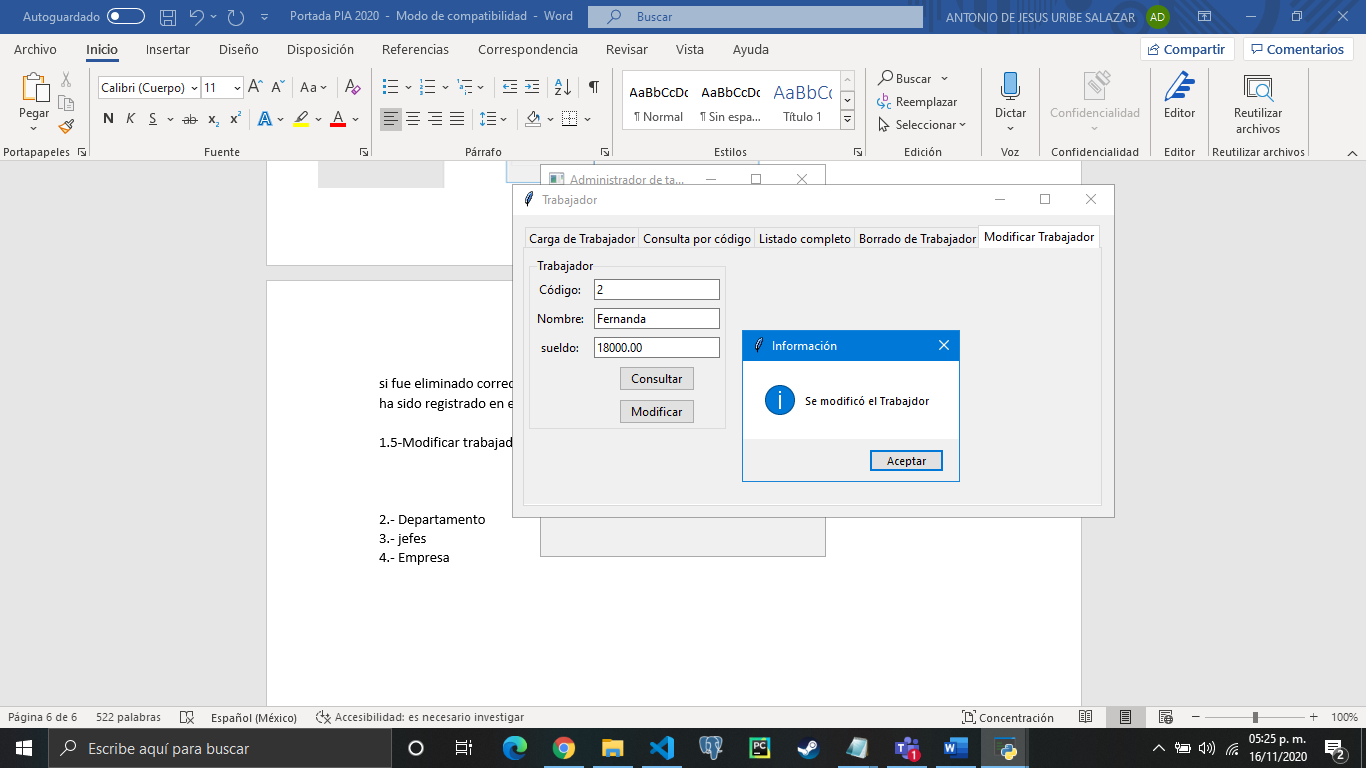
En esta pestaña nos muestra los códigos que captura el sistema y los nombres y sueldo capturados por el usuario. Siempre y cuando el usuario de clic en Listado Completo.

**1.4-Borrador de trabajador**

En este apartado borramos al trabajador de nombre y sueldo, solamente insertado el código establecido por el sistema.

Nota: puede verificar que el código, nombre y sueldo; dirigiéndose a la pestaña Listado completo.

Importante: si se despliega la pestaña de información indicara si fue eliminado correctamente y/o sino saldrá una ventana que no se ha encontrado. Es decir, no ha sido registrado en el sistema.

**1.5-Modificar trabajador**

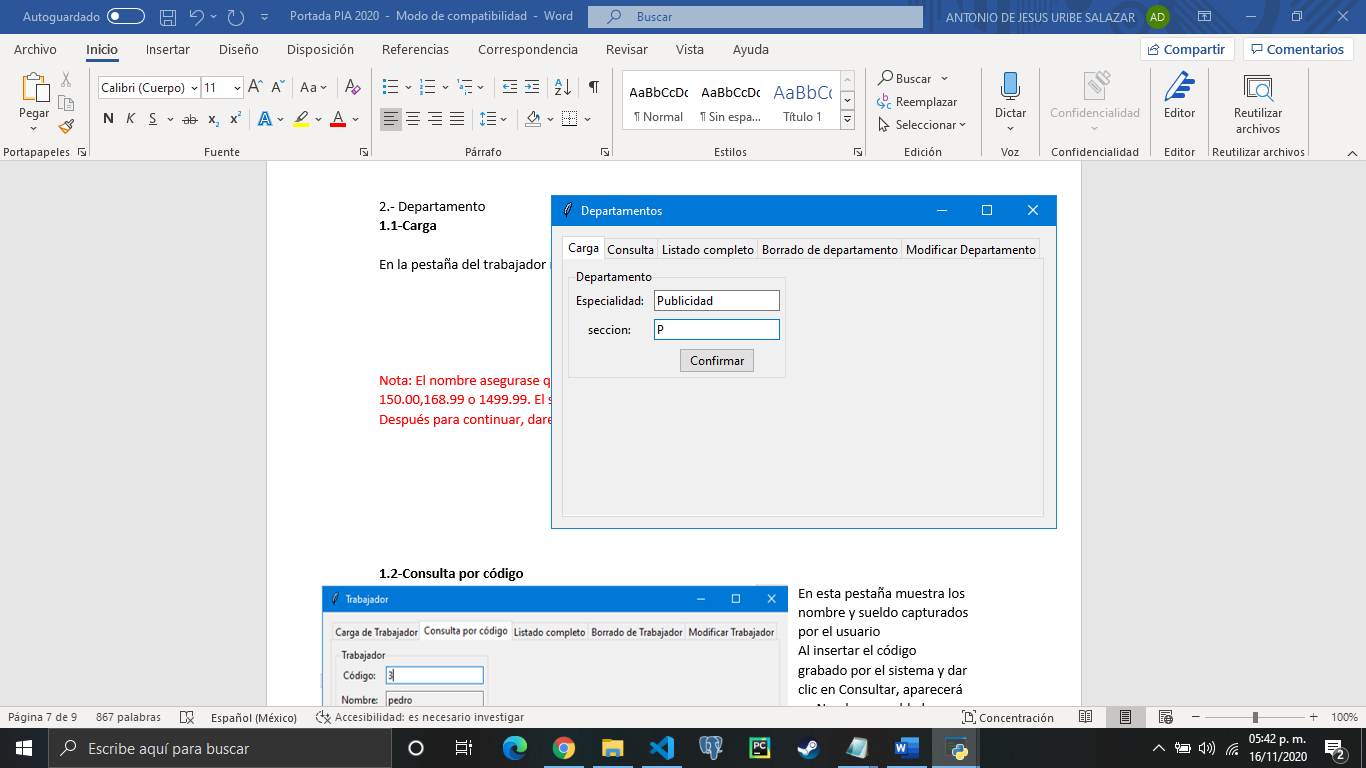
En esta pestaña muestra de modificar o actualizar un usuario. Remplazando sus datos por otros como el ejemplo de arriba. Insertamos el código establecido por el sistema y para modificar. Luego el Nombre y sueldo que fijaremos como nuevos y después clic en Modificar.

Nota: Si quieres saber sobre el trabajador de dicho código puedes darle clic en Consultar y en seguida muestra el Nombre y sueldo que corresponde dicho código.

**2.- Departamento**

**2.1-Carga**

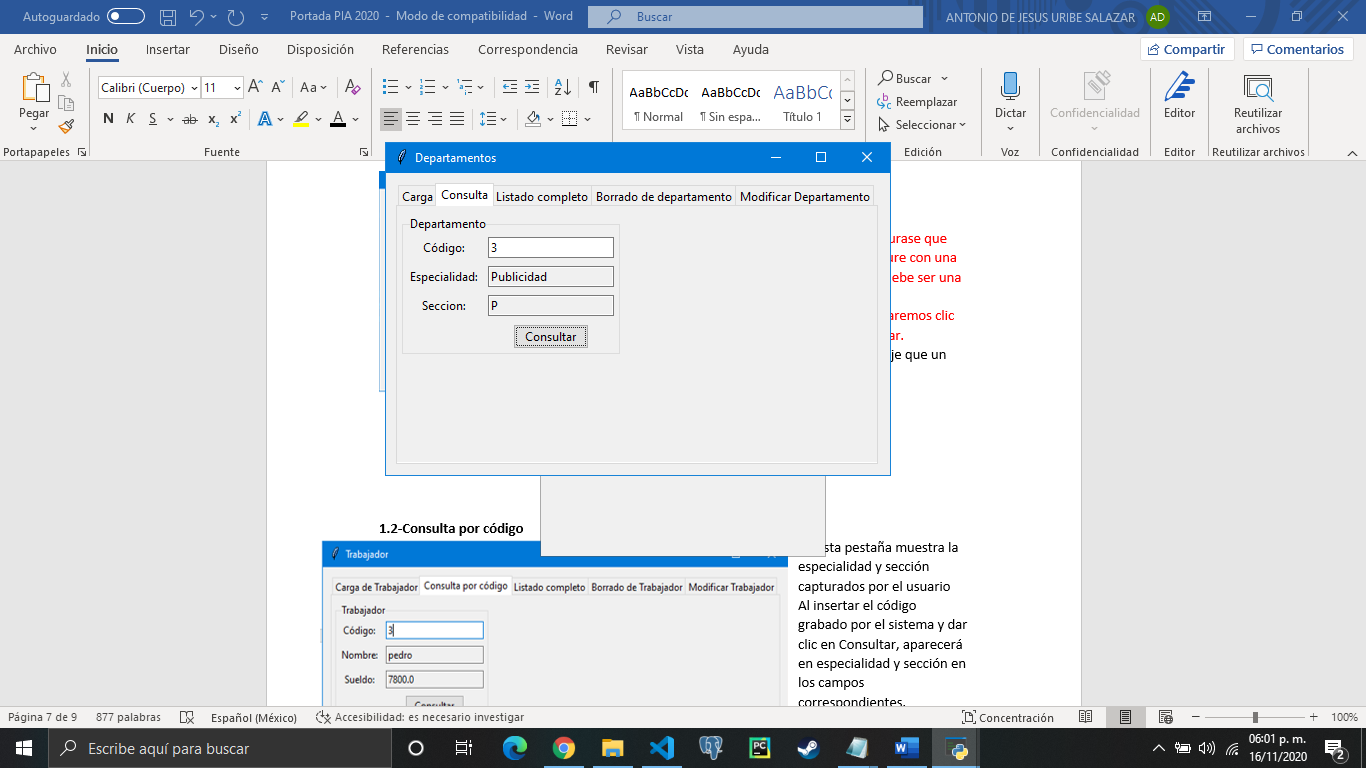
En la pestaña del trabajador nos pide capturar el nombre del trabajador y su sueldo

Nota: La especialidad asegurase que este bien y la sección capture con una Letra del área. La sección debe ser una letra del abecedario.

Después para continuar, daremos clic en Confirmar para continuar.

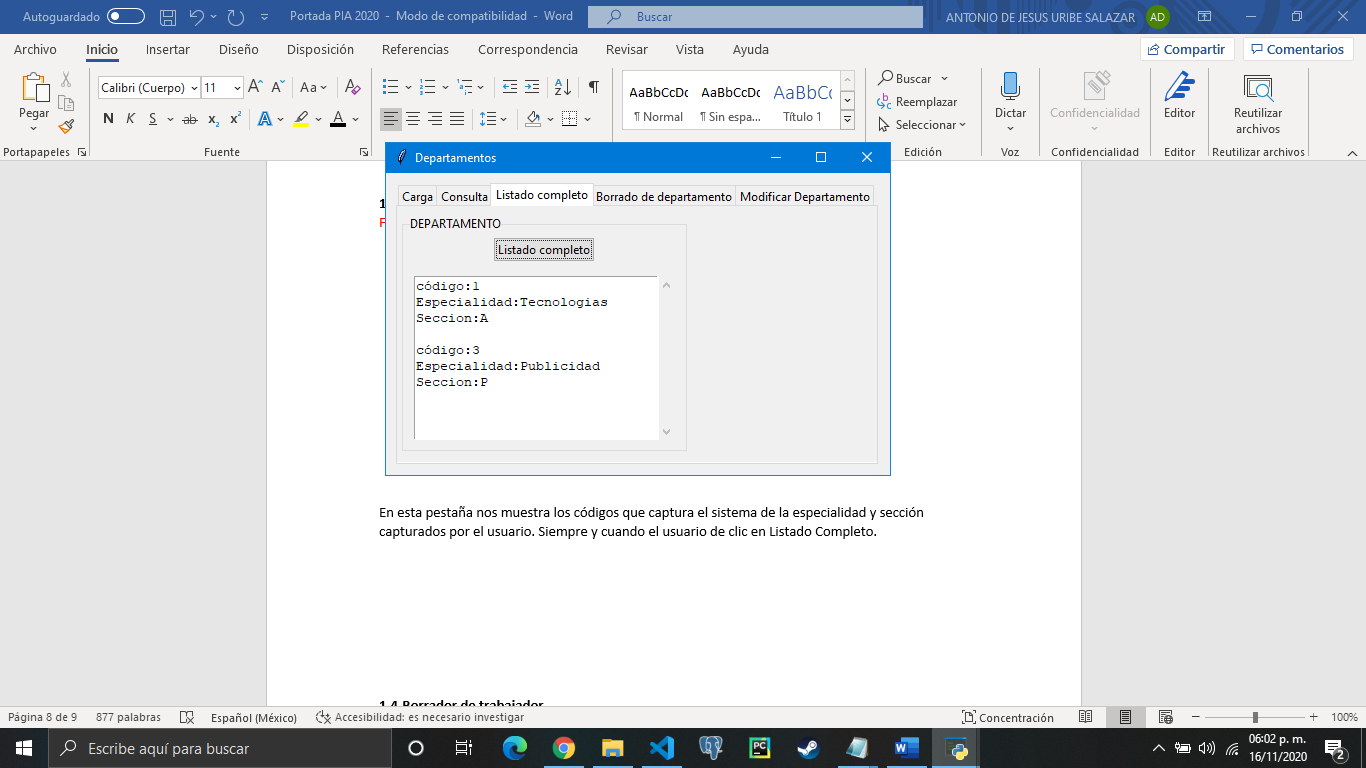
Después cargara un mensaje que un éxito cargarlo.

**2.2-Consulta por código**

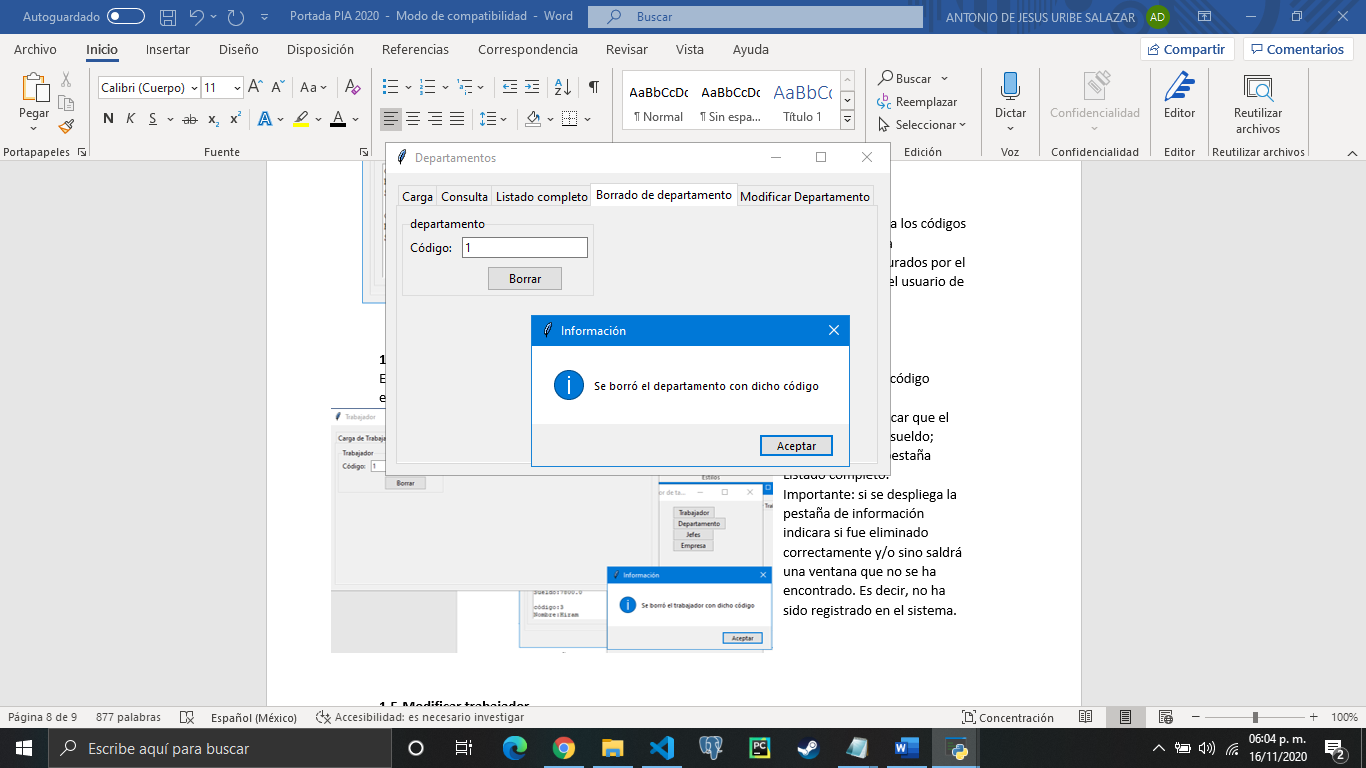
En esta pestaña muestra la especialidad y sección capturados por el usuario

Al insertar el código grabado por el sistema y dar clic en Consultar, aparecerá en especialidad y sección en los campos correspondientes.

**2.3-Listado completo**

Figura listado completo

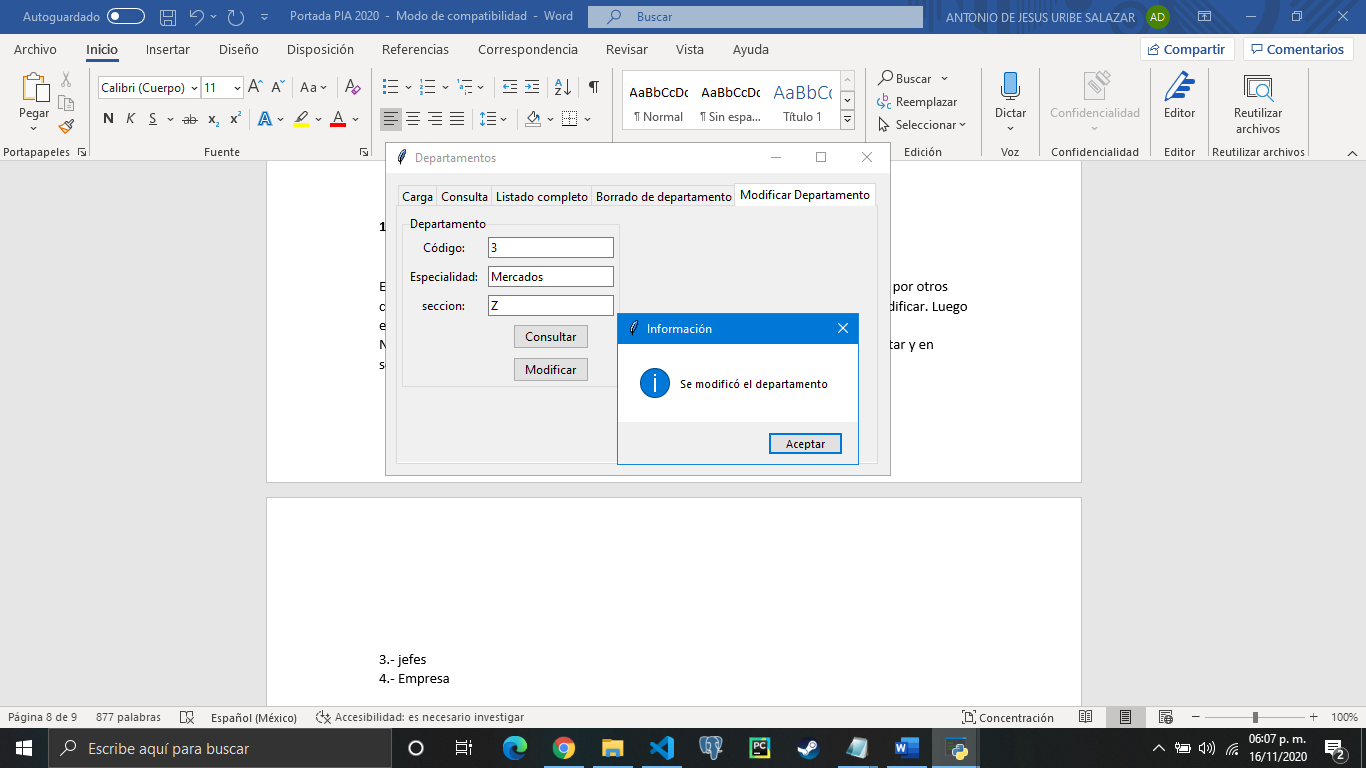
En esta pestaña nos muestra los códigos que captura el sistema de la especialidad y sección capturados por el usuario. Siempre y cuando el usuario de clic en Listado Completo.

**2.4-Borrador de trabajador**

En este apartado borramos al trabajador de especialidad y sección; solamente insertado el código establecido por el sistema.

Nota: puede verificar que el código, **especialidad y sección**; dirigiéndose a la pestaña Listado completo.

Importante: si se despliega la pestaña de información indicara si fue eliminado correctamente y/o sino saldrá una ventana que no se ha encontrado. Es decir, no ha sido registrado en el sistema.

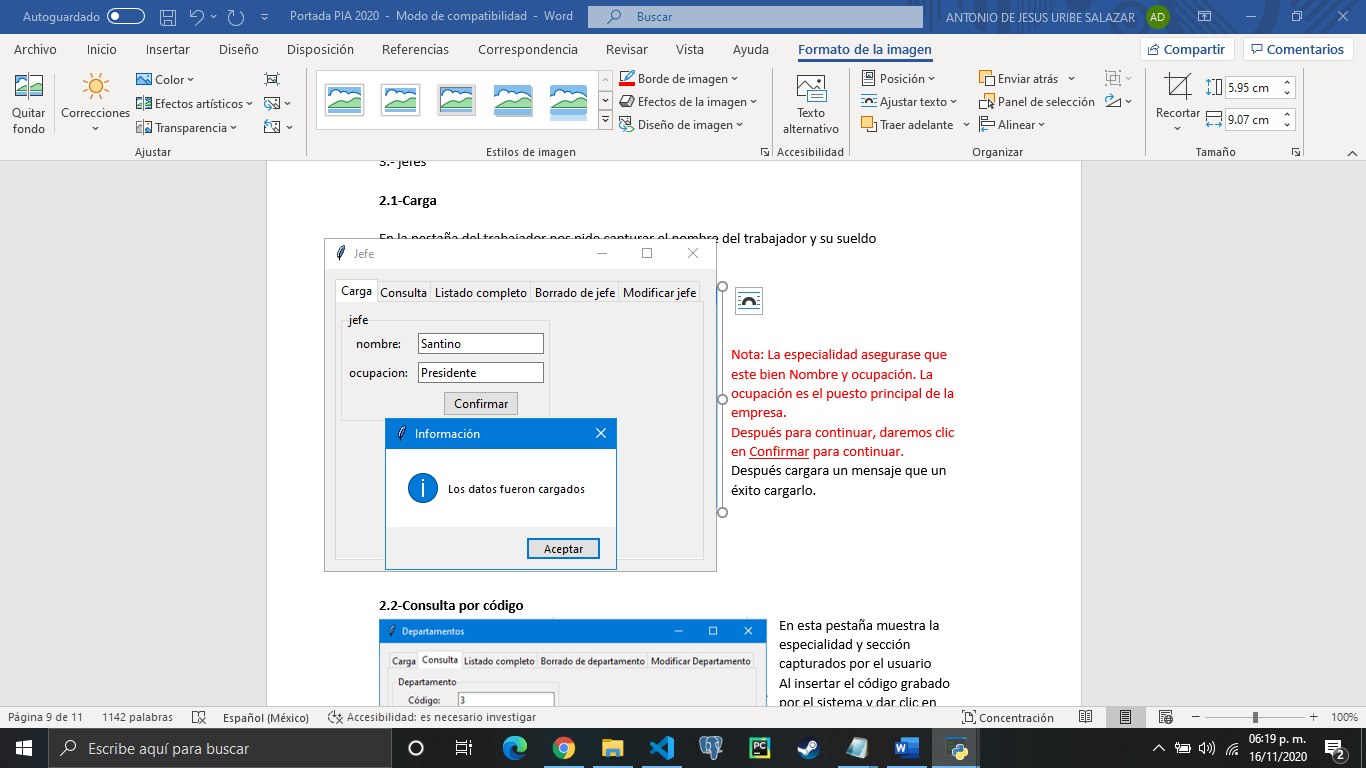
**2.5-Modificar trabajador**

En esta pestaña muestra de modificar o actualizar un usuario. Remplazando sus datos por otros como el ejemplo de arriba. Insertamos el código establecido por el sistema y para modificar. Luego el **especialidad y sección** que fijaremos como nuevos y después clic en Modificar.

Nota: Si quieres saber sobre el trabajador de dicho código puedes darle clic en Consultar y en seguida muestra el especialidad y sección que corresponde dicho código.

**3.- jefes**

**3.1-Carga**

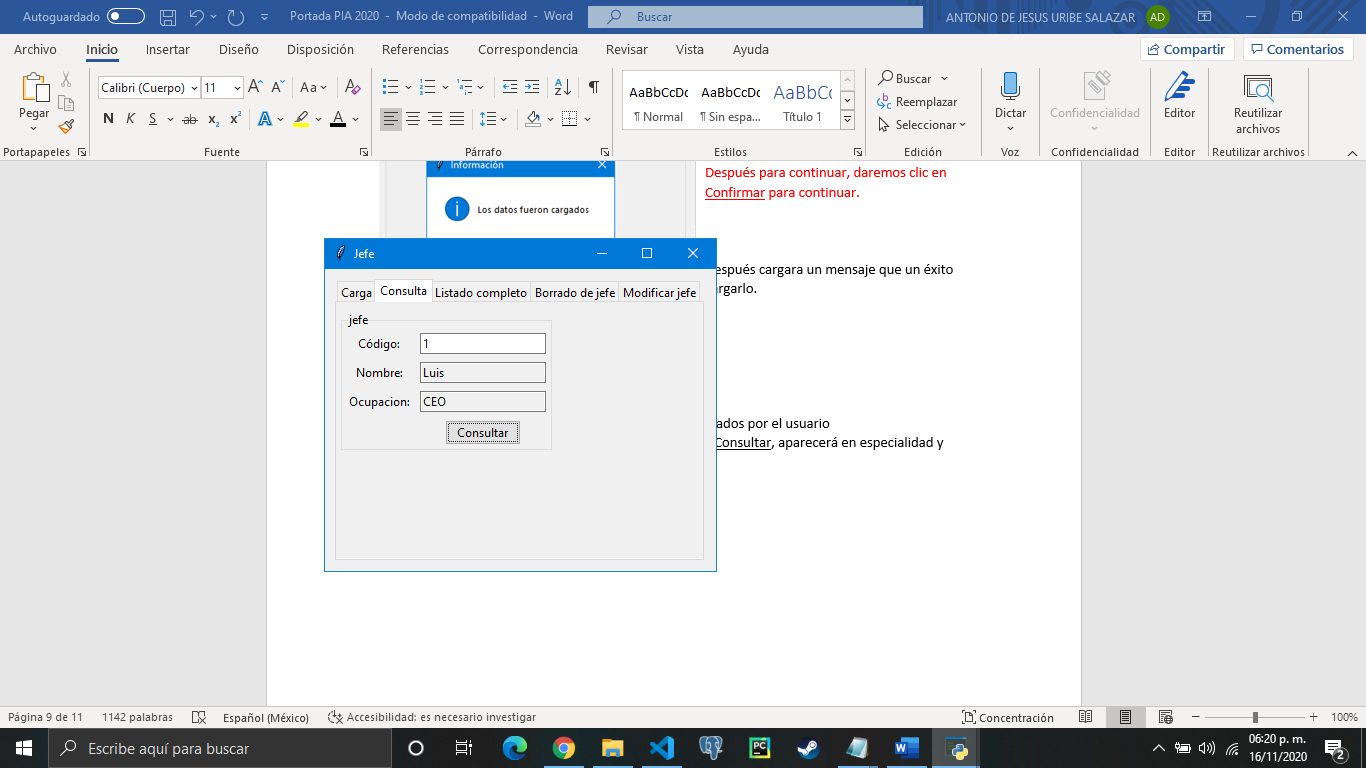
En la pestaña del trabajador nos pide capturar el nombre del trabajador y su sueldo

Nota: La especialidad asegurase que este bien Nombre y ocupación. La ocupación es el puesto principal de la empresa.

Después para continuar, daremos clic en Confirmar para continuar.

Después cargara un mensaje que un éxito cargarlo.

**3.2-Consulta por código**



En esta pestaña muestra la Nombre y ocupación capturados por el usuario

Al insertar el código grabado por el sistema y dar clic en Consultar, aparecerá en Nombre y ocupación en los campos correspondientes.

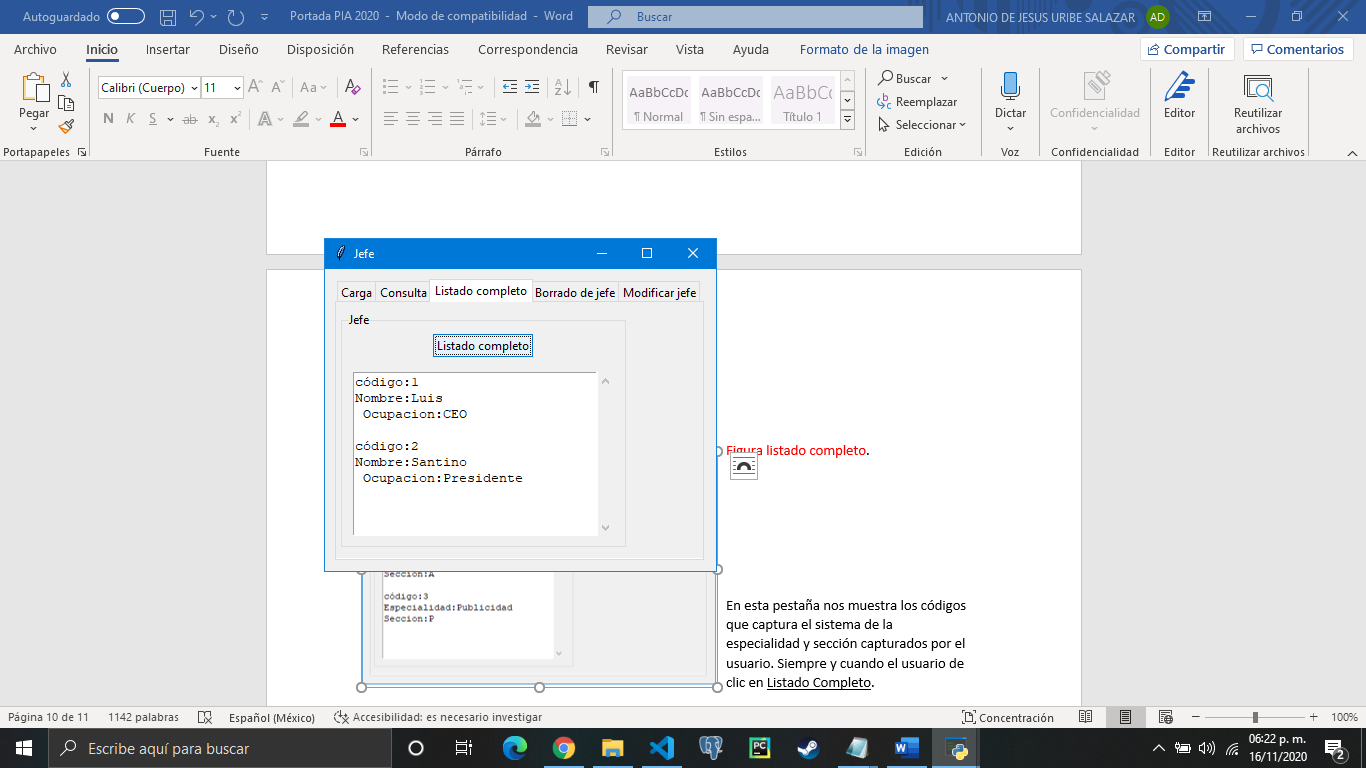
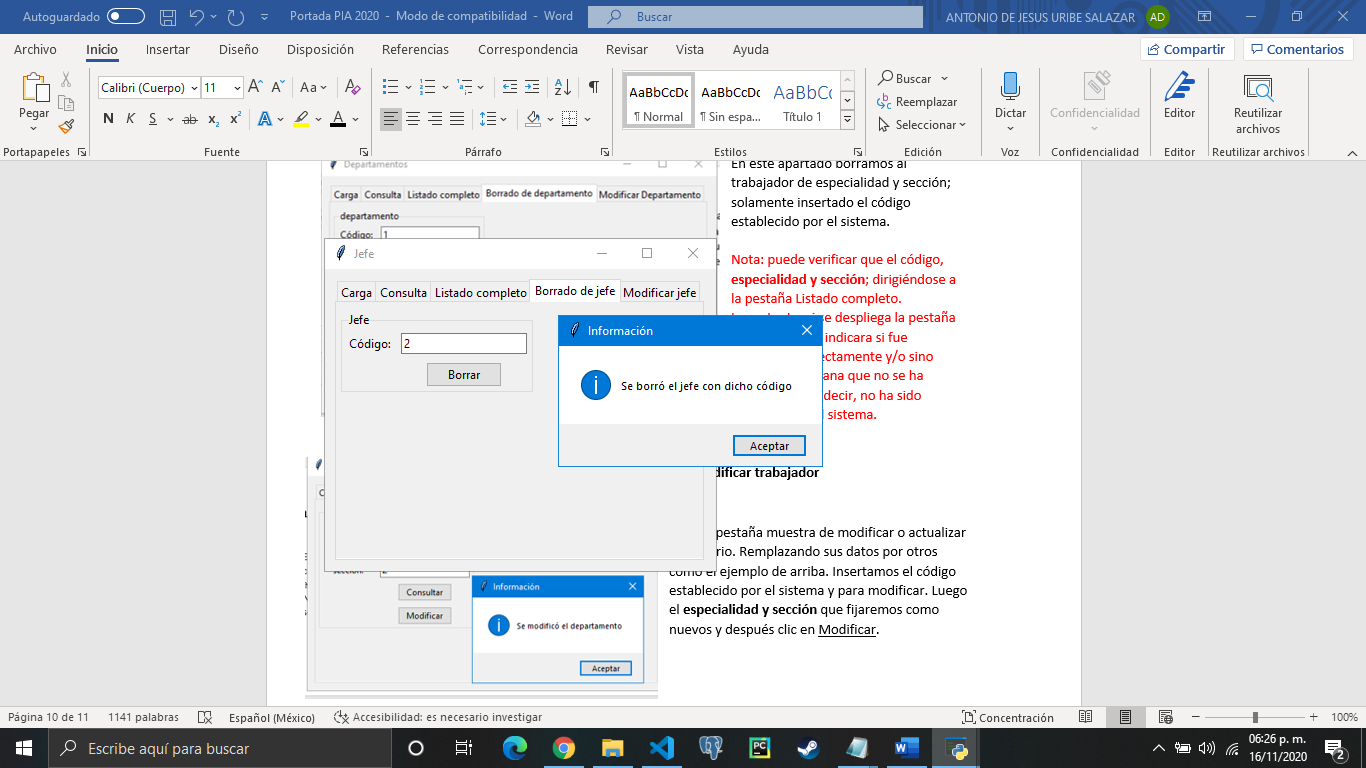
**3.3-Listado completo**

Figura listado completo.

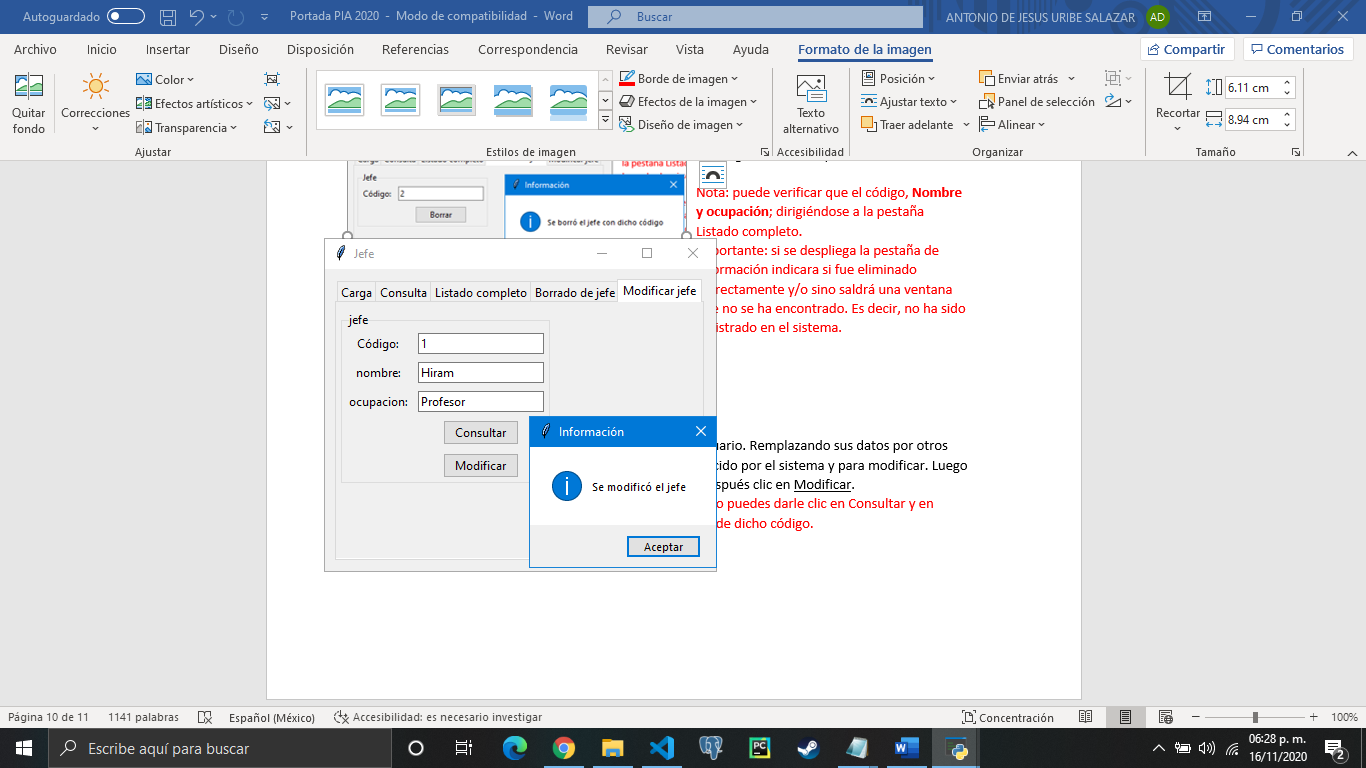
En esta pestaña nos muestra los códigos que captura el sistema de Nombre y ocupación capturados por el usuario. Siempre y cuando el usuario de clic en Listado Completo.

**3.4-Borrado de jefe**

En este apartado borramos al trabajador de especialidad y sección; solamente insertado el código establecido por el sistema.

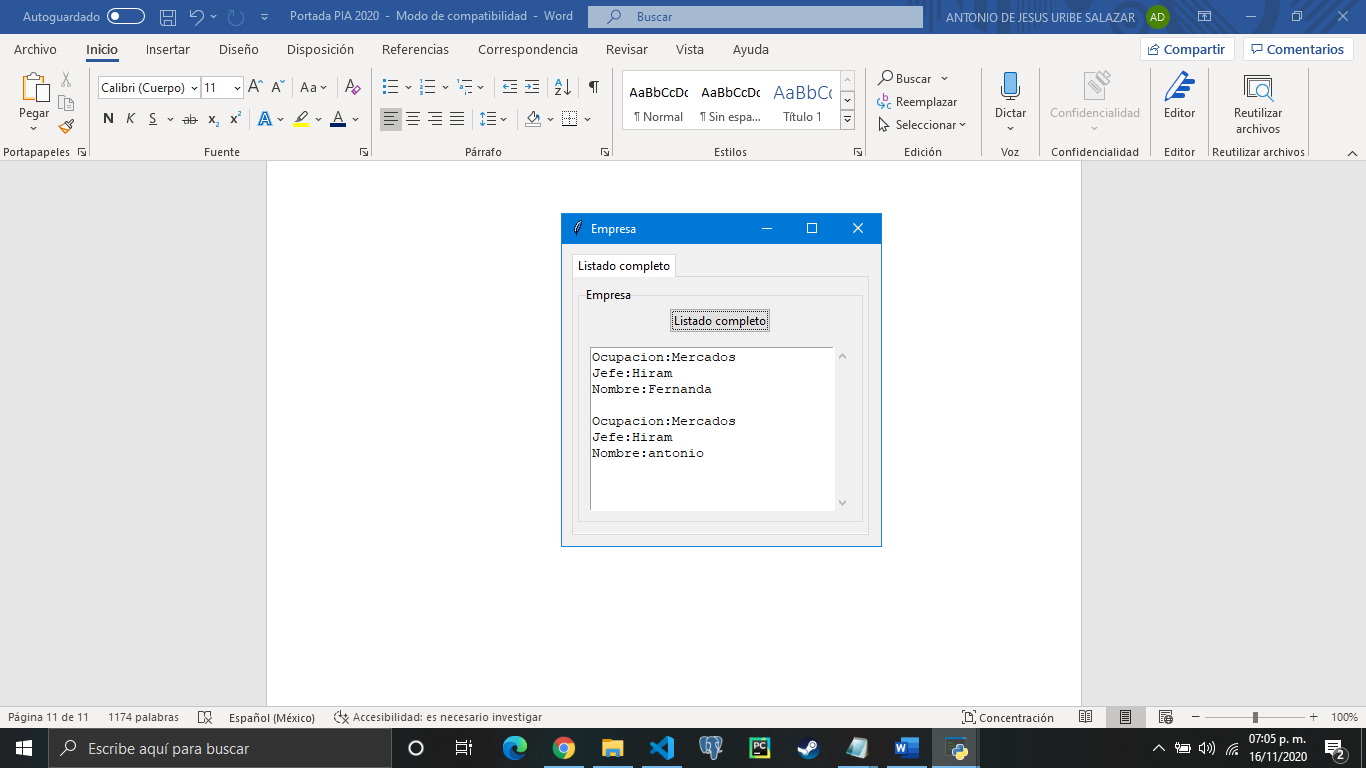
Nota: puede verificar que el código, **Nombre y ocupación**; dirigiéndose a la pestaña Listado completo.

Importante: si se despliega la pestaña de información indicara si fue eliminado correctamente y/o sino saldrá una ventana que no se ha encontrado. Es decir, no ha sido registrado en el sistema.

**3.5-Modificar jefe**

En esta pestaña muestra de modificar o actualizar un usuario. Remplazando sus datos por otros como el ejemplo de arriba. Insertamos el código establecido por el sistema y para modificar. Luego el **Nombre y ocupación** que fijaremos como nuevos y después clic en Modificar.

Nota: Si quieres saber sobre el jefe de dicho código puedes darle clic en Consultar y en seguida muestra el nombre y ocupación que corresponde dicho código.

**4.- Empresa**

Es la consulta total de nuestros registros. Así como el nombre del trabajador, nombre del jefe y la ocupación de cada persona en la Empresa.

**ALGORITMOS:**

**database.py.py**

import sqlite3

conexion = sqlite3.connect("C:/Users/schedule/Desktop/A\_Programas/Semestre/interfaz/bd/prueba1.db")

try:

    conexion.execute("""

create table trabajador(

id\_trabajador integer primary key autoincrement,

nombre text,

sueldo real

)

""")

    print("La tabla se creo correctamente")

except sqlite3.OperationalError:

    print("la tabla articulos ya existe")

conexion.close()

#Colocar las tablas una por una en conexion.execute()

#Tabla para crear Trabajdor

'''

create table trabajador(

id\_trabajador integer primary key autoincrement,

nombre text,

sueldo real

);

'''

#Tabla para crear departamento

'''

create table departamento(

codigo integer primary key autoincrement,

especialidad text

seccion

);

'''

#Tabla para crear Jefes

'''

create table trabajador(

id\_trabajador integer primary key autoincrement,

nombre text,

sueldo real

);

'''

**pruebaD.py**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mb

from tkinter import scrolledtext as st

import Pruebade

class FormularioDepartamento:

    def \_\_init\_\_(self):

        self.departamento1=Pruebade.Departamento()

        self.ventana1=tk.Tk()

        self.ventana1.title("Departamentos")

        self.cuaderno1 = ttk.Notebook(self.ventana1)

        self.carga\_articulos()

        self.consulta\_por\_codigo()

        self.listado\_completo()

        self.borrado()

        self.modificar()

        self.cuaderno1.grid(column=0, row=0, padx=10, pady=10)

        self.ventana1.mainloop()

    def carga\_articulos(self):

        self.pagina1 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina1, text="Carga")

        self.labelframe1=ttk.LabelFrame(self.pagina1, text="Departamento")

        self.labelframe1.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe1, text="Especialidad:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.descripcioncarga=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe1, textvariable=self.descripcioncarga)

        self.entrydescripcion.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe1, text="seccion:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.preciocarga=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe1, textvariable=self.preciocarga)

        self.entryprecio.grid(column=1, row=1, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe1, text="Confirmar", command=self.agregar)

        self.boton1.grid(column=1, row=2, padx=4, pady=4)

    def agregar(self):

        datos=(self.descripcioncarga.get(), self.preciocarga.get())

        self.departamento1.alta(datos)

        mb.showinfo("Información", "Los datos fueron cargados")

        self.descripcioncarga.set("")

        self.preciocarga.set("")

    def consulta\_por\_codigo(self):

        self.pagina2 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina2, text="Consulta")

        self.labelframe2=ttk.LabelFrame(self.pagina2, text="Departamento")

        self.labelframe2.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe2, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigo=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe2, textvariable=self.codigo)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe2, text="Especialidad:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.descripcion=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe2, textvariable=self.descripcion, state="readonly")

        self.entrydescripcion.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe2, text="Seccion:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.precio=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe2, textvariable=self.precio, state="readonly")

        self.entryprecio.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe2, text="Consultar", command=self.consultar)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

    def consultar(self):

        datos=(self.codigo.get(), )

        respuesta=self.departamento1.consulta(datos)

        if len(respuesta)>0:

            self.descripcion.set(respuesta[0][0])

            self.precio.set(respuesta[0][1])

        else:

            self.descripcion.set('')

            self.precio.set('')

            mb.showinfo("Información", "No existe un departamento con dicho código")

    def listado\_completo(self):

        self.pagina3 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina3, text="Listado completo")

        self.labelframe3=ttk.LabelFrame(self.pagina3, text="DEPARTAMENTO")

        self.labelframe3.grid(column=0, row=0, padx=5, pady=10)

        self.boton1=ttk.Button(self.labelframe3, text="Listado completo", command=self.listar)

        self.boton1.grid(column=0, row=0, padx=4, pady=4)

        self.scrolledtext1=st.ScrolledText(self.labelframe3, width=30, height=10)

        self.scrolledtext1.grid(column=0,row=1, padx=10, pady=10)

    def listar(self):

        respuesta=self.departamento1.recuperar\_todos()

        self.scrolledtext1.delete("1.0", tk.END)

        for fila in respuesta:

            self.scrolledtext1.insert(tk.END, "código:"+str(fila[0])+

                                              "\nEspecialidad:"+fila[1]+

                                              "\nSeccion:"+str(fila[2])+"\n\n")

    def borrado(self):

        self.pagina4 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina4, text="Borrado de departamento")

        self.labelframe4=ttk.LabelFrame(self.pagina4, text="departamento")

        self.labelframe4.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe4, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigoborra=tk.StringVar()

        self.entryborra=ttk.Entry(self.labelframe4, textvariable=self.codigoborra)

        self.entryborra.grid(column=1, row=0, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe4, text="Borrar", command=self.borrar)

        self.boton1.grid(column=1, row=1, padx=4, pady=4)

    def borrar(self):

        datos=(self.codigoborra.get(), )

        cantidad=self.departamento1.baja(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se borró el departamento con dicho código")

        else:

            mb.showinfo("Información", "No existe un departamento con dicho código")

    def modificar(self):

        self.pagina5 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina5, text="Modificar Departamento")

        self.labelframe5=ttk.LabelFrame(self.pagina5, text="Departamento")

        self.labelframe5.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe5, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigomod=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe5, textvariable=self.codigomod)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe5, text="Especialidad:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.descripcionmod=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe5, textvariable=self.descripcionmod)

        self.entrydescripcion.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe5, text="seccion:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.preciomod=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe5, textvariable=self.preciomod)

        self.entryprecio.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Consultar", command=self.consultar\_mod)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Modificar", command=self.modifica)

        self.boton1.grid(column=1, row=4, padx=4, pady=4)

    def modifica(self):

        datos=(self.descripcionmod.get(), self.preciomod.get(), self.codigomod.get())

        cantidad=self.departamento1.modificacion(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se modificó el departamento")

        else:

            mb.showinfo("Información", "No existe un departamento con dicho código")

    def consultar\_mod(self):

        datos=(self.codigomod.get(), )

        respuesta=self.departamento1.consulta(datos)

        if len(respuesta)>0:

            self.descripcionmod.set(respuesta[0][0])

            self.preciomod.set(respuesta[0][1])

        else:

            self.descripcionmod.set('')

            self.preciomod.set('')

            mb.showinfo("Información", "No existe un departamento con dicho código")

**pruebade.py**

import sqlite3

class Departamento:

    def abrir(self):

        conexion=sqlite3.connect("bd/prueba1.db")

        return conexion

    def alta(self, datos):

        cone=self.abrir()

        cursor=cone.cursor()

        sql="insert into departamento(especialidad,seccion) values (?,?)"

        cursor.execute(sql, datos)

        cone.commit()

        cone.close()

    def consulta(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select especialidad, seccion from departamento where codigo=?"

            cursor.execute(sql, datos)

            return cursor.fetchall()

        finally:

            cone.close()

    def recuperar\_todos(self):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select codigo, especialidad, seccion from departamento"

            cursor.execute(sql)

            return cursor.fetchall()

        finally:

            cone.close()

    def baja(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="delete from departamento where codigo=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas borradas

        except:

            cone.close()

    def modificacion(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="update departamento set especialidad=?, seccion=? where codigo=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas modificadas

        except:

            cone.close()

class Trabajador:

    def abrir(self):

        conexion=sqlite3.connect("bd/prueba1.db")

        return conexion

    def alta(self, datos):

        cone=self.abrir()

        cursor=cone.cursor()

        sql="insert into trabajador(nombre,sueldo) values (?,?)"

        cursor.execute(sql, datos)

        cone.commit()

        cone.close()

    def consulta(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select nombre, sueldo  from trabajador where id\_trabajador =? "

            cursor.execute(sql, datos)

            return cursor.fetchall()

        finally:

            cone.close()

    def recuperar\_todos(self):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select id\_trabajador,nombre, sueldo from trabajador"

            cursor.execute(sql)

            return cursor.fetchall()

        finally:

            cone.close()

    def baja(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="delete from trabajador where id\_trabajador=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas borradas

        except:

            cone.close()

    def modificacion(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="update trabajador set nombre=?, sueldo=? where id\_trabajador=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas modificadas

        except:

            cone.close()

class Jefes:

    def abrir(self):

        conexion=sqlite3.connect("bd/prueba1.db")

        return conexion

    def alta(self, datos):

        cone=self.abrir()

        cursor=cone.cursor()

        sql="insert into jefe(nombre,ocupacion) values (?,?)"

        cursor.execute(sql, datos)

        cone.commit()

        cone.close()

    def consulta(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select nombre, ocupacion from jefe where id\_jefe=?"

            cursor.execute(sql, datos)

            return cursor.fetchall()

        finally:

            cone.close()

    def recuperar\_todos(self):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select id\_jefe, nombre, ocupacion from jefe"

            cursor.execute(sql)

            return cursor.fetchall()

        finally:

            cone.close()

    def baja(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="delete from jefe where id\_jefe=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas borradas

        except:

            cone.close()

    def modificacion(self, datos):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="update jefe set nombre=?, ocupacion=? where id\_jefe=?"

            cursor.execute(sql, datos)

            cone.commit()

            return cursor.rowcount # retornamos la cantidad de filas modificadas

        except:

            cone.close()

class Empresa:

    def abrir(self):

        conexion=sqlite3.connect("bd/prueba1.db")

        return conexion

    def recuperar\_todos(self):

        try:

            cone=self.abrir()

            cursor=cone.cursor()

            sql="select departamento.especialidad,jefe.nombre, trabajador.nombre from departamento,trabajador,jefe"

            cursor.execute(sql)

            return cursor.fetchall()

        finally:

            cone.close()

**pruebaE.py**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mb

from tkinter import scrolledtext as st

import Pruebade

class FormularioEmpresa:

    def \_\_init\_\_(self):

        self.departamento1=Pruebade.Empresa()

        self.ventana1=tk.Tk()

        self.ventana1.title("Empresa")

        self.cuaderno1 = ttk.Notebook(self.ventana1)

        self.listado\_completo()

        self.cuaderno1.grid(column=0, row=0, padx=10, pady=10)

        self.ventana1.mainloop()

**PruebaJ.py**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mb

from tkinter import scrolledtext as st

import Pruebade

class FormularioJefe:

    def \_\_init\_\_(self):

        self.jefe1=Pruebade.Jefes()

        self.ventana1=tk.Tk()

        self.ventana1.title("Jefe")

        self.cuaderno1 = ttk.Notebook(self.ventana1)

        self.cargajefe()

        self.consulta\_jefe()

        self.listado\_completo()

        self.borrado()

        self.modificar()

        self.cuaderno1.grid(column=0, row=0, padx=10, pady=10)

        self.ventana1.mainloop()

    def cargajefe(self):

        self.pagina1 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina1, text="Carga")

        self.labelframe1=ttk.LabelFrame(self.pagina1, text="jefe")

        self.labelframe1.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe1, text="nombre:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.nombrecarga=tk.StringVar()

        self.entrynombre=ttk.Entry(self.labelframe1, textvariable=self.nombrecarga)

        self.entrynombre.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe1, text="ocupacion:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.ocupacioncarga=tk.StringVar()

        self.entryocupacion=ttk.Entry(self.labelframe1, textvariable=self.ocupacioncarga)

        self.entryocupacion.grid(column=1, row=1, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe1, text="Confirmar", command=self.agregar)

        self.boton1.grid(column=1, row=2, padx=4, pady=4)

    def agregar(self):

        datos=(self.nombrecarga.get(), self.ocupacioncarga.get())

        self.jefe1.alta(datos)

        mb.showinfo("Información", "Los datos fueron cargados")

        self.nombrecarga.set("")

        self.ocupacioncarga.set("")

    def consulta\_jefe(self):

        self.pagina2 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina2, text="Consulta")

        self.labelframe2=ttk.LabelFrame(self.pagina2, text="jefe")

        self.labelframe2.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe2, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigo=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe2, textvariable=self.codigo)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe2, text="Nombre:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.descripcion=tk.StringVar()

        self.entrynombre=ttk.Entry(self.labelframe2, textvariable=self.descripcion, state="readonly")

        self.entrynombre.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe2, text="Ocupacion:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.precio=tk.StringVar()

        self.entryocupacion=ttk.Entry(self.labelframe2, textvariable=self.precio, state="readonly")

        self.entryocupacion.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe2, text="Consultar", command=self.consultar)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

    def consultar(self):

        datos=(self.codigo.get(), )

        respuesta=self.jefe1.consulta(datos)

        if len(respuesta)>0:

            self.descripcion.set(respuesta[0][0])

            self.precio.set(respuesta[0][1])

        else:

            self.descripcion.set('')

            self.precio.set('')

            mb.showinfo("Información", "No existe un jefe con dicho código")

    def listado\_completo(self):

        self.pagina3 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina3, text="Listado completo")

        self.labelframe3=ttk.LabelFrame(self.pagina3, text="Jefe")

        self.labelframe3.grid(column=0, row=0, padx=5, pady=10)

        self.boton1=ttk.Button(self.labelframe3, text="Listado completo", command=self.listar)

        self.boton1.grid(column=0, row=0, padx=4, pady=4)

        self.scrolledtext1=st.ScrolledText(self.labelframe3, width=30, height=10)

        self.scrolledtext1.grid(column=0,row=1, padx=10, pady=10)

    def listar(self):

        respuesta=self.jefe1.recuperar\_todos()

        self.scrolledtext1.delete("1.0", tk.END)

        for fila in respuesta:

            self.scrolledtext1.insert(tk.END, "código:"+str(fila[0])+

                                              "\nNombre:"+fila[1]+

                                              "\n Ocupacion:"+str(fila[2])+"\n\n")

    def borrado(self):

        self.pagina4 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina4, text="Borrado de jefe")

        self.labelframe4=ttk.LabelFrame(self.pagina4, text="Jefe")

        self.labelframe4.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe4, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigoborra=tk.StringVar()

        self.entryborra=ttk.Entry(self.labelframe4, textvariable=self.codigoborra)

        self.entryborra.grid(column=1, row=0, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe4, text="Borrar", command=self.borrar)

        self.boton1.grid(column=1, row=1, padx=4, pady=4)

    def borrar(self):

        datos=(self.codigoborra.get(), )

        cantidad=self.jefe1.baja(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se borró el jefe con dicho código")

        else:

            mb.showinfo("Información", "No existe un jefe con dicho código")

    def modificar(self):

        self.pagina5 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina5, text="Modificar jefe")

        self.labelframe5=ttk.LabelFrame(self.pagina5, text="jefe")

        self.labelframe5.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe5, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigomod=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe5, textvariable=self.codigomod)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe5, text="nombre:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.nombremod=tk.StringVar()

        self.entrynombre=ttk.Entry(self.labelframe5, textvariable=self.nombremod)

        self.entrynombre.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe5, text="ocupacion:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.ocupacionmod=tk.StringVar()

        self.entryocupacion=ttk.Entry(self.labelframe5, textvariable=self.ocupacionmod)

        self.entryocupacion.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Consultar", command=self.consultar\_mod)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Modificar", command=self.modifica)

        self.boton1.grid(column=1, row=4, padx=4, pady=4)

    def modifica(self):

        datos=(self.nombremod.get(), self.ocupacionmod.get(), self.codigomod.get())

        cantidad=self.jefe1.modificacion(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se modificó el jefe")

        else:

            mb.showinfo("Información", "No existe un jefe con dicho código")

    def consultar\_mod(self):

        datos=(self.codigomod.get(), )

        respuesta=self.jefe1.consulta(datos)

        if len(respuesta)>0:

            self.nombremod.set(respuesta[0][0])

            self.ocupacionmod.set(respuesta[0][1])

        else:

            self.nombremod.set('')

            self.ocupacionmod.set('')

            mb.showinfo("Información", "No existe un jefe con dicho código")

**PruebaT.py**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mb

from tkinter import scrolledtext as st

import Pruebade

class FormularioTrabajador:

    def \_\_init\_\_(self):

        self.trabajador1=Pruebade.Trabajador()

        self.ventana1=tk.Tk()

        self.ventana1.title("Trabajador")

        self.cuaderno1 = ttk.Notebook(self.ventana1)

        self.carga\_articulos()

        self.consulta\_por\_codigo()

        self.listado\_completo()

        self.borrado()

        self.modificar()

        self.cuaderno1.grid(column=0, row=0, padx=10, pady=10)

        self.ventana1.mainloop()

    def carga\_articulos(self):

        self.pagina1 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina1, text="Carga de Trabajador")

        self.labelframe1=ttk.LabelFrame(self.pagina1, text="Trabajador")

        self.labelframe1.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe1, text="Nombre:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.descripcioncarga=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe1, textvariable=self.descripcioncarga)

        self.entrydescripcion.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe1, text="Sueldo:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.preciocarga=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe1, textvariable=self.preciocarga)

        self.entryprecio.grid(column=1, row=1, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe1, text="Confirmar", command=self.agregar)

        self.boton1.grid(column=1, row=2, padx=4, pady=4)

    def agregar(self):

        datos=(self.descripcioncarga.get(), self.preciocarga.get())

        self.trabajador1.alta(datos)

        mb.showinfo("Información", "Los datos fueron cargados")

        self.descripcioncarga.set("")

        self.preciocarga.set("")

    def consulta\_por\_codigo(self):

        self.pagina2 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina2, text="Consulta por código")

        self.labelframe2=ttk.LabelFrame(self.pagina2, text="Trabajador")

        self.labelframe2.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe2, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigo=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe2, textvariable=self.codigo)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe2, text="Nombre:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.descripcion=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe2, textvariable=self.descripcion, state="readonly")

        self.entrydescripcion.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe2, text="Sueldo:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.precio=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe2, textvariable=self.precio, state="readonly")

        self.entryprecio.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe2, text="Consultar", command=self.consultar)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

    def consultar(self):

        datos=(self.codigo.get(), )

        respuesta=self.trabajador1.consulta(datos)

        if len(respuesta)>0:

            self.descripcion.set(respuesta[0][0])

            self.precio.set(respuesta[0][1])

        else:

            self.descripcion.set('')

            self.precio.set('')

            mb.showinfo("Información", "No existe un artículo con dicho código")

    def listado\_completo(self):

        self.pagina3 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina3, text="Listado completo")

        self.labelframe3=ttk.LabelFrame(self.pagina3, text="Trabajador")

        self.labelframe3.grid(column=0, row=0, padx=5, pady=10)

        self.boton1=ttk.Button(self.labelframe3, text="Listado completo", command=self.listar)

        self.boton1.grid(column=0, row=0, padx=4, pady=4)

        self.scrolledtext1=st.ScrolledText(self.labelframe3, width=30, height=10)

        self.scrolledtext1.grid(column=0,row=1, padx=10, pady=10)

    def listar(self):

        respuesta=self.trabajador1.recuperar\_todos()

        self.scrolledtext1.delete("1.0", tk.END)

        for fila in respuesta:

            self.scrolledtext1.insert(tk.END, "código:"+str(fila[0])+

                                              "\nNombre:"+fila[1]+

                                              "\nSueldo:"+str(fila[2])+"\n\n")

    def borrado(self):

        self.pagina4 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina4, text="Borrado de Trabajador")

        self.labelframe4=ttk.LabelFrame(self.pagina4, text="Trabajador")

        self.labelframe4.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe4, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigoborra=tk.StringVar()

        self.entryborra=ttk.Entry(self.labelframe4, textvariable=self.codigoborra)

        self.entryborra.grid(column=1, row=0, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe4, text="Borrar", command=self.borrar)

        self.boton1.grid(column=1, row=1, padx=4, pady=4)

    def borrar(self):

        datos=(self.codigoborra.get(), )

        cantidad=self.trabajador1.baja(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se borró el trabajador con dicho código")

        else:

            mb.showinfo("Información", "No existe un trabajador con dicho código")

    def modificar(self):

        self.pagina5 = ttk.Frame(self.cuaderno1)

        self.cuaderno1.add(self.pagina5, text="Modificar Trabajador")

        self.labelframe5=ttk.LabelFrame(self.pagina5, text="Trabajador")

        self.labelframe5.grid(column=0, row=0, padx=5, pady=10)

        self.label1=ttk.Label(self.labelframe5, text="Código:")

        self.label1.grid(column=0, row=0, padx=4, pady=4)

        self.codigomod=tk.StringVar()

        self.entrycodigo=ttk.Entry(self.labelframe5, textvariable=self.codigomod)

        self.entrycodigo.grid(column=1, row=0, padx=4, pady=4)

        self.label2=ttk.Label(self.labelframe5, text="Nombre:")

        self.label2.grid(column=0, row=1, padx=4, pady=4)

        self.descripcionmod=tk.StringVar()

        self.entrydescripcion=ttk.Entry(self.labelframe5, textvariable=self.descripcionmod)

        self.entrydescripcion.grid(column=1, row=1, padx=4, pady=4)

        self.label3=ttk.Label(self.labelframe5, text="sueldo:")

        self.label3.grid(column=0, row=2, padx=4, pady=4)

        self.preciomod=tk.StringVar()

        self.entryprecio=ttk.Entry(self.labelframe5, textvariable=self.preciomod)

        self.entryprecio.grid(column=1, row=2, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Consultar", command=self.consultar\_mod)

        self.boton1.grid(column=1, row=3, padx=4, pady=4)

        self.boton1=ttk.Button(self.labelframe5, text="Modificar", command=self.modifica)

        self.boton1.grid(column=1, row=4, padx=4, pady=4)

    def modifica(self):

        datos=(self.descripcionmod.get(), self.preciomod.get(), self.codigomod.get())

        cantidad=self.trabajador1.modificacion(datos)

        if cantidad==1:

            mb.showinfo("Información", "Se modificó el Trabajdor")

        else:

            mb.showinfo("Información", "No existe un Trabajdor con dicho código")

    def consultar\_mod(self):

        datos=(self.codigomod.get(), )

        respuesta=self.trabajador1.consulta(datos)

        if len(respuesta)>0:

            self.descripcionmod.set(respuesta[0][0])

            self.preciomod.set(respuesta[0][1])

        else:

            self.descripcionmod.set('')

            self.preciomod.set('')

            mb.showinfo("Información", "No existe un Trabajdor con dicho código")

**menu.py**

import PruebaT

import PruebaD

import PruebaJ

import PruebaE

import tkinter as tk

import wx

def trabajador(event):

    a=PruebaT.FormularioTrabajador()

def departamento(event):

    a=PruebaD.FormularioDepartamento()

def jefe(event):

    a=PruebaJ.FormularioJefe()

def empresa(event):

    a=PruebaE.FormularioEmpresa()

app = wx.App()

frame = wx.Frame(None, -1, 'Administrador de tareas.py')

frame.SetDimensions(100,200,300,400)

panel = wx.Panel(frame, wx.ID\_ANY)

labelA = wx.StaticText(panel, wx.ID\_ANY, "Administrador de la empresa", pos=(100,0))

button = wx.Button(panel, wx.ID\_ANY, 'Trabajador', (120, 30),size=(100,50))

button.Bind(wx.EVT\_BUTTON, trabajador)

button2 = wx.Button(panel, wx.ID\_ANY, 'Departamento', (120, 80),size=(100,50))

button2.Bind(wx.EVT\_BUTTON, departamento)

button2.SetBackgroundColour(wx.Colour(200,50,100))

button3 = wx.Button(panel, wx.ID\_ANY, 'Jefes', (120, 130),size=(100,50))

button3.Bind(wx.EVT\_BUTTON, jefe)

button3.SetBackgroundColour(wx.Colour(150,255,100))

button4 = wx.Button(panel, wx.ID\_ANY, 'Empresa', (120, 180),size=(100,50))

button4.Bind(wx.EVT\_BUTTON, empresa)

button4.SetBackgroundColour(wx.Colour(200,255,100))

frame.Show()

frame.Centre()

app.MainLoop()

#aplicacion1=FormularioArticulos()

#a=Prueba2.FormularioTrabajador()

#a=Prueba2.FormularioTrabajador()

**ANECA 4.1 DEMOSTRAR LA CONCIENTIZACIÓN SOBRE LA NECESIDAD DE TENER UNA CONDUCTA ÉTICA Y PROFESIONAL DE PRIMER NIVEL Y CONOCIMIENTOS DE LOS CÓDIGOS DE CONDUCTA PROFESIONAL.**

Nosotros como futuros desarrolladores debemos de tener concientización sobre cumplir con una ética de trabajo, debido a que tenemos acceso a información que otras personas de otras áreas no tienen.

Para realizar este software nosotros lo trabajamos en binas, nosotros quisimos implementar los conocimientos obtenidos en clase y además agregar algunos elementos para que nuestro software reflejara nuestra creatividad y conocimientos del tema.

Es importante recalcar que para realizar esta actividad pusimos en práctica el código de ética establecido por nuestra Facultad, tales como: *honestidad*: al desarrollar nuestro propio trabajo desde cero, sin ayuda de terceras personas o sin plagiar algún trabajo ya realizado; *responsabilidad social:* al implementar los valores éticos y morales; *solidaridad*: debido a que ambos integrantes del equipo trabajamos en conjunto para alcanzar un mismo objetivo; por último, *justicia:* debido a que delegamos las actividades de una manera que la dificultas fuera proporcional.

**“CONCLUSIONES INDIVIDUALES”**

**ESCAMILLA TAPIA FERNANDA YAMILETH:**

La tecnología sin dudar ha beneficiado al ser humano en todos los ámbitos posibles, sin embargo, no debemos de restarle importancia a la programación, ya que, sin ella la tecnología no podría seguir avanzando. Nosotros como futuros desarrolladores tenemos que estar buscando soluciones a problemas que pueden tener en diversas áreas, tal es el caso en este proyecto, nosotros nos percatamos que la gestión en las empresas es muy importante y por eso quisimos dedicarle nuestro proyecto a realizar un programa que cumpliera con este objetivo. Todo esto no lo hubiéramos podido realizar sin la guía de nuestro profesor y las ejemplificaciones que realizo a lo largo del semestre. Desarrollar no es una tarea sencilla, pero practicando podemos mejorar.

**URIBE SALAZAR ANTONIO DE JESUS:**

El uso de las tecnologías para el desarrollo de programas a través de problemas mejora el uso de soluciones de cada usuario o cliente con experiencia en situaciones difíciles que presenta. Es por eso por lo que este programa organice un sistema para mantener el orden de una organización empresarial y gestión para cumplir el registro fácil y rápido para una persona de recursos humanos o administrador. El programa puede cumplir con aspectos básicos para el cliente o solución; e implementarlo en necesidades de registro amplio.

El trabajo de algunas librerías y funciones de Python ayudo a implementar las herramientas básicas de un gestor. Mi aprendizaje se extendió hasta utilizar este programa y ver la solución del problema.

**“CONCLUSIÓN”**

**ESPAÑOL:**

A lo largo de este trabajo nos pudimos dar cuenta de que un software es esencial para cualquier campo laboral, debido a que gracias a él se logra tener un control sobre los aspectos más importantes o lo podemos utilizar como una herramienta.

Elaborar un software no es una tarea sencilla, debido a que conlleva horas de trabajo, ya que nosotros como desarrolladores tenemos que analizar el problema que queremos solucionar para así poder realizarlo. Programar no es una tarea sencilla, en ocasiones nos encontramos con errores en el programa y esto conlleva a tener que volver a analizar para buscar el error, sin embargo, el resultado final de un buen programa puede ayudar a muchas personas y facilitarles su trabajo.

Concentramos todos los conocimientos adquiridos a lo largo de este semestre para poder realizar esta actividad, nuestro objetivo principal era lograr implementar todo lo visto en estos meses y a su vez reflejar nuestra creatividad en el programa. Nosotros consideramos que el programa presentado cumple con el objetivo de ayudar a la administración de una empresa.

**INGLÉS:**

Throughout this work we were able to realize that software is essential for any work field, because thanks to it we can have control over the most important aspects or we can use it as a tool.

Developing software is not an easy task, because it takes hours of work, since we as developers have to analyze the problem, we want to solve in order to do it. Programming is not an easy task, sometimes we find errors in the program and this leads to having to re-analyze to find the error, however, the end result of a good program can help many people and facilitate their work.

We concentrated all the knowledge acquired throughout this semester to be able to carry out this activity, our main objective was to implement everything seen in these months and in turn reflect our creativity in the program. We consider that the presented program meets the objective of helping the administration of a company.