[1]

Prototipo de desarrollo de software PRIH para mejorar el rendimiento del sistema operativo Windows 10 (Julio de 2021)

Juan C. Bernal Autor

Resumen - Desarrollo de un programa python, que ayuda con la mejora del rendimiento y eficiencia del sistema operativo Windows 10, para computadores de gama baja y media, en ordenadores de hogares y pymes, que no hagan parte de una red con políticas de grupo, codificado en python 3.9.2, como una aplicación de escritorio, en unión con una base de datos Oracle, desde la cual se tienen guardados los servicios que pueden ser modificados por los usuarios.

Índice de Términos - Memoria RAM, Memoria Virtual, Registros, Servicios

I. INTRODUCCION

En las últimas décadas la tecnología ha avanzado a pasos agigantados y más personas utilizan un computador para la ayuda en sus trabajos, para algunas tareas en la casa o simplemente para su entretenimiento, por esto es cada vez más normal ver que las personas pasan más tiempo en un computador de escritorio o portátil.

En Colombia los ordenadores cada vez son más caros y vemos que muchas personas y empresas no compran o actualizan su hardware y cada vez sacan software con requerimientos mayores, sobrecargando la memoria RAM y el sistema operativo.

La realización de estos procesos puede ser algo difícil o tedioso para las personas que no tiene ningún conocimiento de soporte de computadores, también muchas personas que trabajan en soporte no saben que se pueden realizar algunas acciones para que el sistema operativo mejore un poco el rendimiento y a veces en la instalación del sistema operativo, pueden dejar procesos innecesarios ejecutándose o servicios que el usuario nunca va a utilizar.

Por esto es necesario el desarrollo de un software que facilite la modificación de algunos registros o servicios, para aumentar el rendimiento del sistema operativo, sin perder la estabilidad ni la seguridad de su ordenador ni de sus datos.

[1]Documento recibido el 30 de julio de 2021. Este trabajo fue apoyado en parte por Maria Teresa Tovar y Carlos Augusto Bernal.

Juan C. Bernal Autor de la fundación universitaria del área andina, e-mail: torjandruida@hotmail.com.

II. MANUAL DEL USUARIO

A. *Ingreso a la aplicación*

El ingreso a la aplicación PRIH se realiza mediante el ejecutable app.exe.

Fig. 1. ejecutable de la aplicación app.exe

B. *Pantalla de validación correcta de sistema operativo*

Pantalla en la cual el programa valida si el sistema operativo es el correcto para poder iniciar la aplicación, si el sistema operativo es Windows 10 muestra el nombre del sistema operativo arriba y los botones del menú para que pueda continuar.

Fig. 2. pantalla de validación correcta

C. *Pantalla de validación errónea de sistema operativo*

Pantalla en la cual el programa valida si el sistema operativo es el correcto para poder iniciar la aplicación, cuando el sistema operativo es incorrecto, muestra un label indicando que no es el sistema operativo correcto y carga el botón de salir.

Fig. 3. pantalla de validación errónea

D. Pantalla de efectos visuales

se ingresa mediante el botón “efectos visuales”, salen 3 RadioButton mediante los cuales el usuario puede seleccionar la opción deseada “dejar que windows elija”,”calidad de gráficos”, “rendimiento de gráficos” y dos botones el primero “aplicar cambios”, “reiniciar” para ejecutar el cambio seleccionado, el segundo para reiniciar el pc y que aplique los cambios.

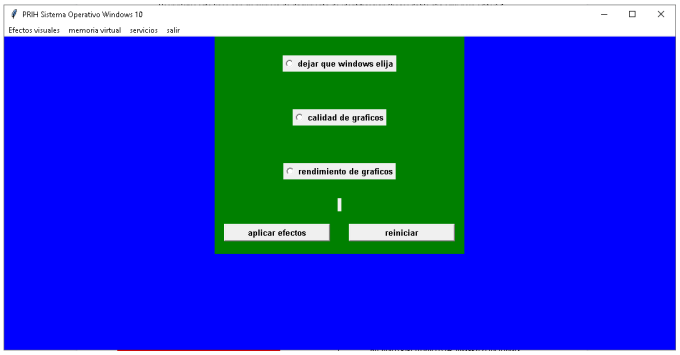

Fig. 4. pantalla de efectos visuales

E. Pantalla de memoria virtual

se ingresa mediante el botón “memoria virtual”, salen 3 RadioButton mediante los cuales el usuario puede seleccionar la opción deseada “administrado por windows”,”mejor rendimiento”, “sin paginación” y dos botones el primero “aplicar memoria”, “reiniciar” para ejecutar el cambio seleccionado, el segundo para reiniciar el pc y que aplique los cambios.

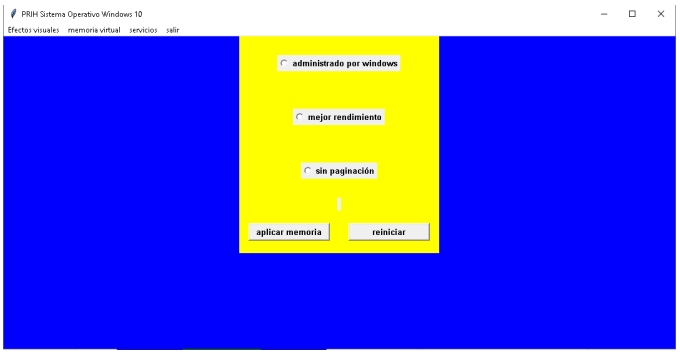

Fig. 5. pantalla de memoria virtual

E. Pantalla de servicios

Se ingresa mediante el botón “servicios’, salen cuadros de selección, el nombre y la descripción del servicio, mediante los cuales el usuario puede seleccionar los servicios a apagar y un botón “apagar servicios seleccionados”en el cual el programa apaga los servicios seleccionados.

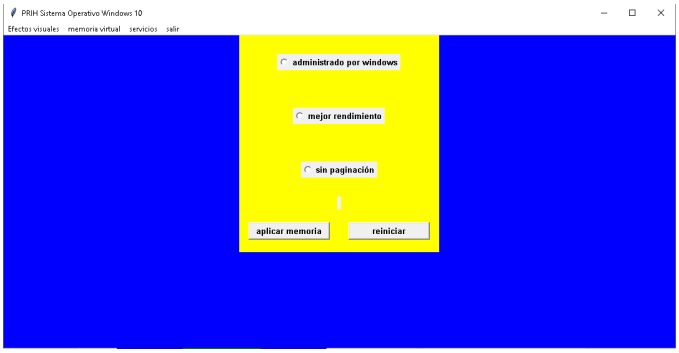

Fig. 5. pantalla de memoria virtual

E. Botón salir

Se utiliza para que el usuario pueda salir del aplicativo.

III. MANUAL DEL PROGRAMADOR

Se presentan los códigos mediante los cuales se realizó la aplicación.

A. Interfaz Gráfica

Código mediante el cual se codificó la interfaz gráfica.

1) Creación de la ventana

```
def main():
    raiz = Tk()
```

2) visualización del frame 1 y ocultación frame 2 y frame 3

```
global ventCont1
        ventCont1 += 1
        miFrame2.pack()
        miFrame3.pack_forget()
        miFrame4.pack_forget()
        miFrame2.config(bg = "green")
        miFrame2.config(width = "1075",height = "350")
        miFrame.destroy()
```

3) verificar la selección del radiobuttom del frame 1

```
selection = " ha seleccionado la opcion " + str(var.get())
            global menuBoton1
            menuBoton1 = var.get()
            label.config(text = selection)
```

4) creación de los radiobuttom del frame 1

```
var = IntVar()
        R1 = Radiobutton(miFrame2, text="dejar que windows elija", variable=var, value=1,font = "none 10 bold", command = sel)
        R1.pack(side = TOP,padx=15, pady=30)

                R2 = Radiobutton(miFrame2, text="calidad de graficos", variable=var, value=2,font = "none 10 bold", command = sel)
        R2.pack(side = TOP, padx=15, pady=30)

            R3 = Radiobutton(miFrame2, text="rendimiento de graficos", variable=var, value=3,font = "none 10 bold", command = sel)
        R3.pack(side = TOP, padx=15, pady=30)
```

5) creación de los botones del frame 1

```
button1 = Button(miFrame2,text = "aplicar efectos", width = "20",height = "1", font = "none 10 bold",
                                                    command=lambda: efectosVisuales.eleccion(menuBoton1) )
        button1.pack(side= LEFT, padx=15, pady=20)

         button2 = Button(miFrame2,text = "reiniciar", width = "20",height = "1", font = "none 10 bold",
                                                    command=lambda: subprocess.run("shutdown -r"))
        button2.pack(side= RIGHT, padx=15, pady=20)
```

6) visualización del frame 2 y ocultación frame 1 y frame 3

```
global ventCont2
      ventCont2 += 1
      miFrame3.pack()
      miFrame2.pack_forget()
      miFrame4.pack_forget()
      miFrame3.config(bg = "yellow")
      miFrame3.config(width = "1075",height = "500")
      miFrame.destroy()
```

7) verificar la selección del radiobuttom del frame 2

```
selection = "You selected the option " + str(var.get())
            global menuBoton2
            menuBoton2 = var.get()
            label.config(text = selection)
```

8) creación de los radiobuttom del frame 2

```
var = IntVar()
            R4 = Radiobutton(miFrame3, text="administrado por windows", variable=var, value=1,font = "none 10 bold", command = sel)
            R4.pack(side = TOP,padx=15, pady=30)

                    R5 = Radiobutton(miFrame3, text="mejor rendimiento", variable=var, value=2,font = "none 10 bold", command = sel)
            R5.pack(side = TOP, padx=15, pady=30)

             R6 = Radiobutton(miFrame3, text="sin paginación", variable=var, value=3,font = "none 10 bold", command = sel)
            R6.pack(side = TOP, padx=15, pady=30)

            label = Label(miFrame3)
            label.pack()
```

9) creación de los botones del frame 2

```
button3 = Button(miFrame3,text = "aplicar memoria", width = "15",height = "1", font = "none 10 bold",
                                                    command=lambda: manejoMemoria.manMem(menuBoton2))
        button3.pack(side= LEFT, padx=15, pady=20)

         button4 = Button(miFrame3,text = "reiniciar", width = "15",height = "1", font = "none 10 bold",
                                                    command=lambda: subprocess.run("shutdown -r"))
        button4.pack(side= RIGHT, padx=15, pady=20)
```

10) visualización del frame 3 y ocultación frame 1 y frame2

```
ventCont3 += 1
      miFrame4.pack()
      miFrame2.pack_forget()
      miFrame3.pack_forget()
      miFrame4.config(bg = "black")
      miFrame4.config(width = "1075",height = "500")
      miFrame.destroy()
```

11) verificación de las selecciones del checkbox

```
for key, value in intvar_dict.items():
            if value.get() > 0:
                print('seleccionada:', key[2])
                seleccion_list.append(key[2])

            servicios.serviciosVer(seleccion_list)
            seleccion_list.clear()

        intvar_dict.clear()
```

```
        for cb in checkbutton_list:
            cb.destroy()
            checkbutton_list.clear()
```

12) creación de los checkbutton y labels

```
    for filename in basDat.fetchData():

                intvar_dict[filename] = IntVar()

                    c = Checkbutton(miFrame4, text=filename[1],
variable=intvar_dict[filename])
                c.grid(sticky=W,row=filename[0], column=0)

                checkbutton_list.append(c)
                        label = Label(miFrame4,text=filename[3],
justify=LEFT)
                label.config(height=1, width=106)
                label.grid(sticky=S, row=filename[0], column=1)

            numCol = (len(basDat.fetchData()))

            label = Label(miFrame4,text="")
            label.config(height=1, width=3)

            label.grid(row=numCol+1, column=0)
            label.grid(row=numCol+1, column=1)
```

12) configuración de la ventana

```
raiz.title("PRIH "+"Sistema Operativo "+platform.system()+"
"+platform.release())
    raiz.resizable(True,True)
    raiz.geometry("1075x500")
    raiz.config(bg = "blue")
```

13)configuración del frame inicial

```
miFrame = Frame()
    miFrame.pack()
    miFrame.config(bg = "red")
    miFrame.config(width = "1075",height = "500")
```

14) validación de sistema operativo

```
if platform.system() == "Windows" and platform.release() ==
"10":
        barraMenu = Menu(raiz)
            barraMenu.add_command(label = "Efectos visuales",
command=efecVis)
            barraMenu.add_command(label = "memoria virtual",
command=memoVi)
                barraMenu.add_command(label = "servicios",
command=serv)
                barraMenu.add_command(label = "salir",
command=raiz.destroy)
        raiz.config(menu = barraMenu)
```

15) botón salir cuando el sistema operativo es el incorrecto

```
button = Button(miFrame,text = "Salir", width = "10",height =
"1", font = "none 10 bold",command= miFrame.quit)
        button.pack(side= BOTTOM, padx=15, pady=20)
         miLabel2 = Label(miFrame, text = "Su sistema operativo
no es compatible",font = "none 14 bold",anchor =
CENTER,width = "1075",height = "500",bg = "red")
        miLabel2.pack()
```

B. Servicios

Código mediante el cual se codificó la clase Servicios

1)código mediante el cual se realiza el control de los servicios seleccionados

```
lst = lista
    if not lst:
            messagebox.showerror("Error", "no ha seleccionado
ningun servicio")
    else:
        servSel =""
        for serv in lst:

            servSel += " net Start "+serv+" &"
            try:
                s = psutil.win_service_get(serv)
            except:
                print("el servicio no exite")
            print("estado")
            print(s.status())
        manejoServicios.manServ(servSel)
```

C. basDat

Código mediante el cual se codificó la clase basDat

1) *código mediante el cual se realiza la conexión a la base de datos.*

```
host = "localhost"
    user = "SYSTEM"
    passw = "Orcl12345"
    tsname = "ORCL"

    try:
                connection = cx_Oracle.connect(user, passw,
host+"/"+tsname)
    except Exception as error:
        print("no se pudo conectar a la base de datos " + error)
```

```
    else:
        print("conexion realizada")

    return connection
```

2) *código mediante el cual se realiza se cargan los datos de la base de datos*

```
connection = getConnection()
    cursor = connection.cursor()
    sql_fetch_date = "select * from servicios"
    cursor.execute(sql_fetch_date)
    listaTupla = []
    for result in cursor:
        listaTupla.append(result)

    connection.commit()
    cursor.close()
    return listaTupla
```

3) *código mediante el cual se realiza se insertan los datos de la base de datos*

```
connection = getConnection()
    cursor = connection.cursor()
                p1 = "insert into servicios values (3,'camara','camara','fotografica')"
    cursor.execute(p1)
    connection.commit()
    cursor.close()
    print("servicio agregado")
```

4) *código mediante el cual se actualizan los datos de la base de datos*

```
connection = getConnection()
    cursor = connection.cursor()
     sql_update = "update servicios set descripcion='camara alta resolucion' where id_servicios=3"
    cursor.execute(sql_update)
    connection.commit()
    cursor.close()
    print("servicio actualizado")
```

5) *código mediante el cual se borran los datos de la base de datos*

```
connection = getConnection()
    cursor = connection.cursor()
    sql_delete = "delete from servicios where id_servicios=3"
    cursor.execute(sql_delete)
    connection.commit()
    cursor.close()
    print("servicio borrado")
```

D. efectosVisuales

Código mediante el cual se codificó la clase efectosVisuales

1) creación de strings de registros de acuerdo a la selección del usuario para su posterior manejo

```
    print("entro efectos")
    print(boton)
    print(type(boton))
                                                path1 = "SOFTWARE\\Microsoft\\windows\\CurrentVersion\\Explorer\\VisualEffects"
    path2 = "Control Panel\\Desktop"

    reg1 = "VisualFXSetting"
    reg2 = "UserPreferencesMask"

    val1 = b'\x9E\x1E\x07\x80\x12\x00\x00\x00'
    val2 = b'\x9E\x3E\x07\x80\x12\x00\x00\x00'
    val3 = b'\x90\x12\x03\x80\x10\x00\x00\x00'

    if boton == 1:
        print("eligio 1")
        manejoRegistros.set_Registro(path1,reg1,0)
        manejoRegistros.set_Registro(path2,reg2,val1)
    elif boton == 2:
        print("eligio 2")
        manejoRegistros.set_Registro(path1,reg1,1)
        manejoRegistros.set_Registro(path2,reg2,val2)
    elif boton == 3:
        print("eligio 3")
        manejoRegistros.set_Registro(path1,reg1,2)
        manejoRegistros.set_Registro(path2,reg2,val3)
```

E. manejoRegistros

Código mediante el cual se codificó la clase manejoRegistos

1) manejo de registros de acuerdo a la selección y cambio del registro

```
with winreg.ConnectRegistry(None, winreg.HKEY_CURRENT_USER) as hkey:
                with winreg.OpenKey(hkey, path, 0,winreg.KEY_ALL_ACCESS) as sub_key:
            existing_path_value = winreg.EnumValue(sub_key,0)
            if nombre == "VisualFXSetting":
                winreg.SetValueEx(sub_key, "VisualFXSetting", 0, winreg.REG_DWORD, valor)
            elif nombre == "UserPreferencesMask" :
                                winreg.SetValueEx(sub_key, "UserPreferencesMask", 0, winreg.REG_BINARY, valor)
            winreg.CloseKey(sub_key)
            print(type(sub_key))
```

```
        HWND_BROADCAST = 0xFFFF
        WM_SETTINGCHANGE = 0x1A

        SMTO_ABORTIFHUNG = 0x0002

        result = ctypes.c_long()
                                    SendMessageTimeoutW =
ctypes.windll.user32.SendMessageTimeoutW
        SendMessageTimeoutW(
           HWND_BROADCAST,
           WM_SETTINGCHANGE,
           0,
           u"Environment",
           SMTO_ABORTIFHUNG,
           5000,
           ctypes.byref(result))

        print(f"{existing_path_value}")
        print(type(existing_path_value))
```

F. manejoMemoria

Código mediante el cual se codificó la clase manejoMemoria

1) verificación de cuanta memoria libre tiene el disco duro

```
mem = virtual_memory()
disk_usage = psutil.disk_usage("C:\\")
myRoundNumber = math.ceil(mem.total/1024/1024/1024)
ddlibre = int(math.ceil(disk_usage.free/1024/1024/1024))
minimo = str(myRoundNumber*1024)
maximo = str(myRoundNumber*1024*2)
ddMin = int(30 + (myRoundNumber*2))
sizeMem = 'wmic pagefileset where name="C:\\\pagefile.sys"
set InitialSize='+minimo+', MaximumSize='+maximo
```

2) validación del tipo de cuenta de usuario

```
try:
      return ctypes.windll.shell32.IsUserAnAdmin()

   except:
      return False
```

3) cambio de la memoria manejada automaticamente

```
try:
            os.system('cmd /k "wmic computersystem where
name="%computername%"                                    set
AutomaticManagedPagefile=true"')

      except:
         print("no ha ejecutado el comando")
```

4) validación de espacio en el disco duro y selección de rendimiento

```
if ddMin < ddlibre :
        try:
           print("entro4")
            os.system('cmd /k "wmic pagefile list /format:list &
wmic computersystem where name="%computername%" set
AutomaticManagedPagefile=false & '+sizeMem+'"')

        except:
           print("no ha ejecutado el comando")

     else:
          messagebox.showerror("Error", "el sistema no tiene
espacio suficiente")
```

5) trabajar sin memoria virtual

```
   try:
          os.system('cmd /k "wmic pagefile list /format:list &
wmic computersystem where name="%computername%" set
AutomaticManagedPagefile=false & wmic pagefileset where
name="C:\\\pagefile.sys" delete"')

        except:
           print("no ha ejecutado el comando")
```

5) elevación de cuenta de usuario a administrador

```
if is_admin():
      chanMem(opcbtn)

   else:
          ctypes.windll.shell32.ShellExecuteW(None, "runas",
sys.executable, " ".join(sys.argv), None, 1)
```

G. manejoServicios

Código mediante el cual se codificó la clase manejoServicios

1) validación del tipo de cuenta de usuario

```
try:
      return ctypes.windll.shell32.IsUserAnAdmin()

   except:
      return False
```

2) apagado de servicios seleccionados

```
   try:
        print("entro")
        os.system('cmd /k "'+opc+' exit"')


      except:
        print("no ha ejecutado el comando")
```

3) elevación de cuenta de usuario a administrador

```
if is_admin():
      chanServ(opcbtn)

   else:
            ctypes.windll.shell32.ShellExecuteW(None, "runas",
sys.executable, " ".join(sys.argv), None, 1)
      sys.exit(0)


try:
              os.system('cmd /k "wmic computersystem where
name="%computername%" set
AutomaticManagedPagefile=true"')

      except:
        print("no ha ejecutado el comando")
```

IV. CONCLUSIÓN

Se ve que mediante varias librerías de python, se puede realizar una fácil gestión, de los registros y servicios, en el sistema operativo Windows 10, pudiendo así ser modificados, a gusto del usuario, para mejorar el rendimiento de este, por medio de una interfaz gráfica de fácil acceso, que mediante unos botones se pueden cambiar de forma rápida las configuraciones requeridas, sin eliminar o quitar la seguridad del sistema operativo.

REFERENCES

[1] G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.