



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA



Universidad de las Fuerzas Armadas ESPE

Departamento de Ciencias de la Computación

Carrera: Ingeniería en Tecnologías de la
Información

Materia: Ingeniería de Software II

NRC 10523

Proyecto Extra

Alumno: Llumiquinga Llumiquinga Luis Miguel

Tutor: Ing. Efraín Fonseca Carrera

Fecha: 20 de agosto del 2023

Tabla de contenido

Manual de Usuario: Instalación de PyCharm y Python	3
Requisitos Previos	3
Pasos de Instalación	3
Descargar PyCharm	3
Instalar PyCharm	3
Descargar e Instalar Python.....	3
Verificación de la Instalación	4
Manual de Usuario: Instalación de MySQL y Uso desde la Línea de Comandos.....	5
Requisitos Previos	5
Pasos de Instalación	5
Descargar MySQL	5
Instalar MySQL	5
Uso desde la Línea de Comandos.....	6
Creación y Ejecución del Aplicativo Gestión de Estudiantes	8
Paso 1: Creación del Proyecto.....	8
Paso 2: Creación de Archivos	8
Paso 3: Copia y Pega el Código.....	8
Paso 4: Configuración de Bibliotecas	14
Paso 5: Ejecución de los Archivos	14

Manual de Usuario: Instalación de PyCharm y Python

Este manual proporciona instrucciones detalladas para instalar la última versión de PyCharm y el lenguaje Python en tu sistema. PyCharm es un entorno de desarrollo integrado (IDE) popular para trabajar con Python, y Python es el lenguaje de programación que se utiliza en el IDE.

Requisitos Previos

Antes de comenzar con la instalación, asegúrate de que tu sistema cumpla con los siguientes requisitos:

- **Sistema Operativo:** PyCharm y Python son compatibles con Windows, macOS y Linux.
- **Espacio en Disco:** Se recomienda disponer de al menos 1 GB de espacio en disco para la instalación.

Pasos de Instalación

Sigue estos pasos para instalar PyCharm y Python en tu sistema:

Descargar PyCharm

- Accede al sitio web oficial de PyCharm en <https://www.jetbrains.com/pycharm/>.
- Selecciona la versión de PyCharm que deseas descargar (por ejemplo, "Community" o "Professional").
- Haz clic en el botón de descarga correspondiente a tu sistema operativo (Windows, macOS o Linux).

Instalar PyCharm

En Windows:

- Abre el archivo descargado (por ejemplo, pycharm-community-XXXX.X.X.exe).
- El asistente de instalación se abrirá. Sigue las instrucciones en pantalla.
- Elige la ubicación de instalación y las opciones de configuración según tus preferencias.
- Una vez completada la instalación, PyCharm estará listo para usar. Puedes encontrarlo en el menú de inicio.

En macOS:

- Abre el archivo descargado (por ejemplo, pycharm-community-XXXX.X.X.dmg).
- Arrastra y suelta el ícono de PyCharm en la carpeta de "Aplicaciones".
- Abre la carpeta de "Aplicaciones" y haz doble clic en el ícono de PyCharm para iniciar la aplicación.

En Linux:

- Extrae el contenido del archivo descargado (por ejemplo, pycharm-community-XXXX.X.X.tar.gz) en una ubicación deseada.
- Abre la carpeta extraída y ejecuta el archivo bin/pycharm.sh.
- Sigue las instrucciones en pantalla para realizar la configuración inicial.

Descargar e Instalar Python

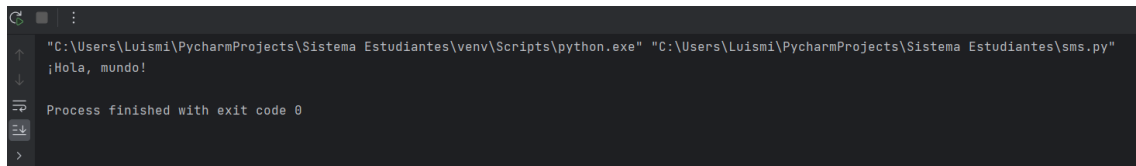
- Accede al sitio web oficial de Python en <https://www.python.org/>.

- En la página de inicio, haz clic en el enlace "Downloads".
- Descarga la última versión estable de Python haciendo clic en el botón "Download Python 3.11.4".
- Ejecuta el archivo descargado y sigue las instrucciones del instalador.
- Asegúrate de marcar la casilla "Add Python X.X to PATH" durante la instalación para que Python sea accesible desde la línea de comandos.

Verificación de la Instalación

Para verificar que tanto PyCharm como Python se han instalado correctamente:

- Abre PyCharm desde el menú de inicio o la carpeta de aplicaciones, según tu sistema operativo.
- Crea un nuevo proyecto o abre un proyecto existente.
- En el editor de código, escribe y ejecuta un pequeño programa en Python para confirmar que la instalación de Python funciona correctamente. Por ejemplo: `printf("!Hola, mundo!")`



```
"C:\Users\Luismi\PycharmProjects\Sistema Estudiantes\venv\Scripts\python.exe" "C:\Users\Luismi\PycharmProjects\Sistema Estudiantes\sms.py"
¡Hola, mundo!
Process finished with exit code 0
```

Has completado la instalación de PyCharm y Python en tu sistema. Ahora estás listo para comenzar a desarrollar aplicaciones en Python utilizando el potente entorno de desarrollo proporcionado por PyCharm.

Recuerda que tanto PyCharm como Python son herramientas en constante desarrollo, por lo que es recomendable mantenerlos actualizados para acceder a las últimas características y mejoras de seguridad.

Manual de Usuario: Instalación de MySQL y Uso desde la Línea de Comandos

Este manual proporciona instrucciones paso a paso para instalar MySQL en tu sistema y cómo utilizarlo desde la línea de comandos para administrar bases de datos y ejecutar consultas.

Requisitos Previos

Antes de comenzar con la instalación, asegúrate de que tu sistema cumpla con los siguientes requisitos:

- **Sistema Operativo:** MySQL es compatible con Windows, macOS y Linux.
- **Espacio en Disco:** Debes disponer de suficiente espacio para la instalación y las bases de datos.

Pasos de Instalación

Sigue estos pasos para instalar MySQL en tu sistema:

Descargar MySQL

- Accede al sitio web oficial de MySQL en <https://dev.mysql.com/downloads/>.
- Selecciona la versión adecuada para tu sistema operativo (por ejemplo, MySQL Community Server).
- Descarga el instalador según tu sistema operativo (Windows, macOS o Linux).

Instalar MySQL

En Windows:

- Ejecuta el instalador descargado (por ejemplo, mysql-installer-community-XXXX.X.X.msi).
- Selecciona "Developer Default" o personaliza la instalación según tus necesidades.
- Configura la contraseña del usuario "root" para la base de datos. Asegúrate de recordarla.
- Completa la instalación siguiendo las instrucciones en pantalla.

En macOS:

- Descarga y abre el archivo de instalación (por ejemplo, mysql-XXXX.X.X-macos10.X.dmg).
- Arrastra y suelta el ícono de MySQL en la carpeta "Aplicaciones".
- Abre la carpeta de "Aplicaciones" y ejecuta el programa "MySQL Workbench".

En Linux:

- Descarga y ejecuta el archivo de instalación (por ejemplo, mysql-apt-config-XXXX.X-X_all.deb) para configurar el repositorio MySQL.
- Instala MySQL usando el sistema de gestión de paquetes de tu distribución. Por ejemplo, en Ubuntu:

```
sudo apt-get update
sudo apt-get install mysql-server
```

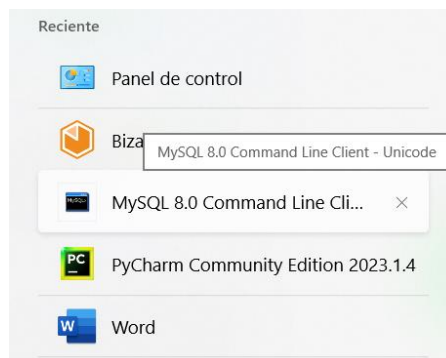
- Durante la instalación, se te pedirá configurar una contraseña para el usuario "root".

Uso desde la Línea de Comandos

Una vez instalado MySQL, puedes usar la línea de comandos para administrar bases de datos y ejecutar consultas. Sigue estos pasos:

Iniciar el Servidor MySQL

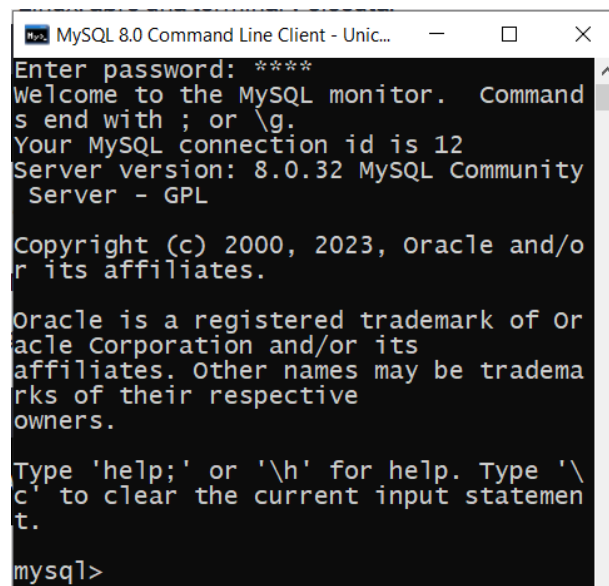
- En Windows, abre el "MySQL Command Line Client" desde el menú de inicio.



- En macOS y Linux, abre una terminal y ejecuta:

```
mysql -u root -p
```

- Te pedirá la contraseña que configuraste durante la instalación.



Crear una nueva base de Datos

Ejecuta las siguientes líneas de código SQL para crear la base de datos a utilizar en el proyecto

```
-- Creación de la base de datos
CREATE DATABASE IF NOT EXISTS gestion_estudiantes;

-- Use the database
USE gestion_estudiantes;

-- Create the student table
CREATE TABLE IF NOT EXISTS student (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(30),
  mobile VARCHAR(10),
  email VARCHAR(30),
  address VARCHAR(100),
  gender VARCHAR(20),
  dob VARCHAR(20),
  date VARCHAR(50),
  time VARCHAR(50)
);
```

Has completado la instalación de MySQL y la base de datos a utilizar en el programa de gestión de estudiantes.

Creación y Ejecución del Aplicativo Gestión de Estudiantes

Paso 1: Creación del Proyecto

1. Abre PyCharm después de la instalación.
2. En la pantalla de inicio, selecciona "Create New Project".
3. En la ventana emergente "New Project", asegúrate de que el entorno Python esté configurado correctamente (puedes usar el entorno predeterminado).
4. En "Location", selecciona la ubicación donde deseas guardar el proyecto. Por ejemplo, puedes crear una carpeta "Sistema Estudiantes" en tu directorio de trabajo y seleccionarla.
5. Haz clic en "Create" para crear el proyecto.

Paso 2: Creación de Archivos

1. En el explorador de proyectos a la izquierda, haz clic derecho en la carpeta del proyecto y selecciona "New" > "Python File".
2. Nombra el archivo como login.py y haz clic en "OK". Repite este proceso para crear otro archivo llamado sms.py.

Paso 3: Copia y Pega el Código

En el proyecto "Sistema Estudiantes" copia las imágenes que se encuentra en este directorio en la carpeta "Imágenes" con el nombre: bd.jpg, logo.png, password.png, sudent.png y user.png. Para su correcta visualización.

Abre el archivo login.py en el editor y copia el código proporcionado para ese archivo. Luego, pega el código en el editor.

login.py

```
from tkinter import *
from tkinter import messagebox
from PIL import ImageTk

def login():
    if usernameEntry.get()==' ' or passwordEntry.get()==' ':
        messagebox.showerror('ADVERTENCIA', 'Campos incorrectos')
    elif usernameEntry.get()=='Faizan' and passwordEntry.get()=='123':
        messagebox.showinfo('BIENVENIDO', 'Campos Correctos')
        window.destroy()
        import sms

    else:
        messagebox.showerror('ADVERTENCIA', 'Ingrese datos correctos')

window = Tk()

window.geometry('1280x700+0+0')      #tamano
window.title('Sistema de Ingreso Estudiantes')

window.resizable(False, False)      #bloqueo

backgroundImage = ImageTk.PhotoImage(file='bg.jpg')

bgLabel=Label(window, image=backgroundImage)
bgLabel.place(x=0, y=0)

loginFrame=Frame(window, bg='white')
```



```

loginFrame.place(x=400, y=150)

logoImage=PhotoImage(file='logo.png')

logoLabel=Label(loginFrame, image=logoImage)
logoLabel.grid(row=0, column=0, columnspan=2, pady=10)
usernameImage=PhotoImage(file='user.png')
usernameLabel=Label(loginFrame, image=usernameImage, text='Usuario', compound=LEFT,
font=('times new roman', 20, 'bold'), bg='white')
usernameLabel.grid(row=1, column=0, pady=10, padx=20)

usernameEntry=Entry(loginFrame, font=('times new roman', 20, 'bold'), bd=5,
fg='royalblue')
usernameEntry.grid(row=1, column=1, pady=10, padx=20)

passwordImage=PhotoImage(file='password.png')
passwordLabel=Label(loginFrame, image=passwordImage, text='Contraseña', compound=LEFT,
font=('times new roman', 20, 'bold'), bg='white')
passwordLabel.grid(row=2, column=0, pady=10, padx=20)

passwordEntry=Entry(loginFrame, font=('times new roman', 20, 'bold'), bd=5,
fg='royalblue')
passwordEntry.grid(row=2, column=1, pady=10, padx=20)

loginButton=Button(loginFrame, text='Login', font=('times new roman', 20, 'bold'),
width=15, fg='white', bg='cornflowerblue', activebackground='cornflowerblue',
activeforeground='white', cursor='hand2', command=login)
loginButton.grid(row=3, column=1, pady=10)
window.mainloop()

```

Haz lo mismo para el archivo sms.py. Copia y pega el código correspondiente en el editor de sms.py.

sms.py

```

from tkinter import *
import ttkthemes
from tkinter import ttk, messagebox, filedialog
import time
import pymysql
import pandas

#funtionality Part

def iexit():
    result=messagebox.askyesno('CONFIRMACION', 'Desea salir?')
    if result:
        root.destroy()
    else:
        pass

def export_data():
    url=filedialog.asksaveasfilename(defaultextension='.csv')
    indexing=studenTable.get_children()
    newlist=[]
    for index in indexing:
        content=studenTable.item(index)
        datalist=content['values']
        newlist.append(datalist)

    table=pandas.DataFrame(newlist, columns=['Id', 'Nombre', 'Teléfono', 'Email',
'Dirección', 'Género', 'Compleaños', 'Fecha Agregado', 'Hora Agregado'])
    table.to_csv(url, index=False)
    messagebox.showinfo('NOTIFICACION', 'Datos exportados')

def toplevel_data(tittle, button_text, command):
    global idEntry, phoneEntry, nameEntry, emailEntry, addressEntry, genderEntry,
dobEntry, screen
    screen = Toplevel()
    screen.title(tittle)
    screen.resizable(False, False)

    idLabel = Label(screen, text='Id', font=('times new roman', 20, 'bold'))
    idLabel.grid(row=0, column=0, padx=30, pady=15, sticky=W)
    idEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
    idEntry.grid(row=0, column=1, pady=15, padx=10)

```

```

nameLabel = Label(screen, text='Nombre', font=('times new roman', 20, 'bold'))
nameLabel.grid(row=1, column=0, padx=30, pady=15, sticky=W)
nameEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
nameEntry.grid(row=1, column=1, pady=15, padx=10)

phoneLabel = Label(screen, text='Teléfono', font=('times new roman', 20, 'bold'))
phoneLabel.grid(row=2, column=0, padx=30, pady=15, sticky=W)
phoneEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
phoneEntry.grid(row=2, column=1, pady=15, padx=10)

emailLabel = Label(screen, text='Email', font=('times new roman', 20, 'bold'))
emailLabel.grid(row=3, column=0, padx=30, pady=15, sticky=W)
emailEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
emailEntry.grid(row=3, column=1, pady=15, padx=10)

addressLabel = Label(screen, text='Dirección', font=('times new roman', 20, 'bold'))
addressLabel.grid(row=4, column=0, padx=30, pady=15, sticky=W)
addressEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
addressEntry.grid(row=4, column=1, pady=15, padx=10)

genderLabel = Label(screen, text='Género', font=('times new roman', 20, 'bold'))
genderLabel.grid(row=5, column=0, padx=30, pady=15, sticky=W)
genderEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
genderEntry.grid(row=5, column=1, pady=15, padx=10)

dobLabel = Label(screen, text='Fecha de cumpleaños', font=('times new roman', 20,
'bold'))
dobLabel.grid(row=6, column=0, padx=30, pady=15, sticky=W)
dobEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)
dobEntry.grid(row=6, column=1, pady=15, padx=10)

student_button = ttk.Button(screen, text=button_text, command=command)
student_button.grid(row=7, columnspan=2, pady=15)

if tittle=='Actualizar Estudiante':
    indexing = studenTable.focus()
    content = studenTable.item(indexing)
    listdata = content['values']
    idEntry.insert(0, listdata[0])
    nameEntry.insert(0, listdata[1])
    phoneEntry.insert(0, listdata[2])
    emailEntry.insert(0, listdata[3])
    addressEntry.insert(0, listdata[4])
    genderEntry.insert(0, listdata[5])
    dobEntry.insert(0, listdata[6])

def update_data():
    query='update student set name=%s, mobile=%s, email=%s, address=%s, gender=%s,
dob=%s, date=%s, time=%s where id=%s'
    mycursor.execute(query, (nameEntry.get(), phoneEntry.get(), emailEntry.get(),
addressEntry.get(), genderEntry.get(), dobEntry.get(), date, currenttime,
idEntry.get()))
    con.commit()
    messagebox.showinfo('ADVERTENCIA', f'Estudiante {idEntry.get()} fue actualizado',
parent= screen)
    screen.destroy()
    show_student()

def show_student():
    query = 'select * from student'
    mycursor.execute(query)
    fetched_data = mycursor.fetchall()
    studenTable.delete(*studenTable.get_children())
    for data in fetched_data:
        studenTable.insert('', END, values=data)

def delete_student():
    indexing=studenTable.focus()
    content=studenTable.item(indexing)
    content_id=content['values'][0]
    query='delete from student where id=%s'
    mycursor.execute(query, content_id)
    con.commit()
    messagebox.showinfo('ELIMINAR', f'Estudiante {content_id} eliminado')
    query='select * from student'
    mycursor.execute(query)

```

```

        fetched_data=mycursor.fetchall()
        studenTable.delete(*studenTable.get_children())
        for data in fetched_data:
            studenTable.insert('', END, values=data)

def search_data():
    query='select *from student where id=%s or name=%s or email=%s or mobile=%s or
address=%s or gender=%s or dob=%s'
    mycursor.execute(query, (idEntry.get(), nameEntry.get(), emailEntry.get(),
phoneEntry.get(), addressEntry.get(), genderEntry.get(), dobEntry.get()))
    studenTable.delete(*studenTable.get_children())
    fetched_data=mycursor.fetchall()
    for data in fetched_data:
        studenTable.insert('', END, values=data)

def add_data():
    if idEntry.get()==' ' or nameEntry.get()==' ' or phoneEntry.get()==' ' or
emailEntry.get()==' ' or addressEntry.get()==' ' or genderEntry.get()==' ' or
dobEntry.get()==' ':
        messagebox.showerror('ADVERTENCIA', 'Los campos son necesarios', parent=screen)
    else:
        try:
            query='insert into student values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'
            mycursor.execute(query, (idEntry.get(), nameEntry.get(), phoneEntry.get(),
emailEntry.get(), addressEntry.get(), genderEntry.get(), dobEntry.get(), date,
currenttime))
            con.commit()
            result=messagebox.askyesno('ADVERTENCIA', 'Estudiante agregado con éxito.
Desea limpiar el formulario?', parent=screen)
            if result:
                idEntry.delete(0, END)
                nameEntry.delete(0, END)
                phoneEntry.delete(0, END)
                emailEntry.delete(0, END)
                addressEntry.delete(0, END)
                genderEntry.delete(0, END)
                dobEntry.delete(0, END)
            else:
                pass

        except:
            messagebox.showerror('ADVERTENCIA', 'Id repetida', parent=screen)
            return

        query='select *from student'
        mycursor.execute(query)
        fetched_data=mycursor.fetchall()
        studenTable.delete(*studenTable.get_children())

        for data in fetched_data:
            studenTable.insert('', END, values=data)

def connect_database():
    def connect():
        global mycursor, con
        try:
            con=pymysql.connect(host='localhost', user='root', password='4189')
            mycursor=con.cursor()

        except:
            messagebox.showerror('Error', 'Datos invalidos', parent=connectWindow)
            return

        try:
            query='create database gestion_estudiantes'
            mycursor.execute(query)
            query='use gestion_estudiantes'
            mycursor.execute(query)
            query='create table student(id int not null primary key, name varchar(30),
mobile varchar(10), email varchar(30), address varchar(100), gender varchar(20), dob
varchar(20), date varchar(50), time varchar(50))'
            mycursor.execute(query)

        except:
            query='use gestion_estudiantes'
            mycursor.execute(query)
            messagebox.showinfo('Exito', 'Conexion exitosa', parent=connectWindow)
            connectWindow.destroy()
            addstudentsButton.config(state=NORMAL)

```

```

searchstudentsButton.config(state=NORMAL)
updatestudentsButton.config(state=NORMAL)
showstudentsButton.config(state=NORMAL)
exportstudentsButton.config(state=NORMAL)
deletestudentsButton.config(state=NORMAL)

connectWindow=Toplevel()
connectWindow.grab_set()
connectWindow.geometry('470x250+730+230')
connectWindow.title('Conexión a la Base de Datos')
connectWindow.resizable(0,0)

hostnameLabel=Label(connectWindow, text='Host Name', font=('arial', 20, 'bold'))
hostnameLabel.grid(row=0, column=0, padx=20)

hostEntry=Entry(connectWindow, font=('roman', 15, 'bold'), bd=2)
hostEntry.grid(row=0, column=1, padx=40, pady=20)

usernameLabel = Label(connectWindow, text='User Name', font=('arial', 20, 'bold'))
usernameLabel.grid(row=1, column=0, padx=20)

usernameEntry = Entry(connectWindow, font=('roman', 15, 'bold'), bd=2)
usernameEntry.grid(row=1, column=1, padx=40, pady=20)

passwordLabel = Label(connectWindow, text='Password', font=('arial', 20, 'bold'))
passwordLabel.grid(row=2, column=0, padx=20)

passwordEntry = Entry(connectWindow, font=('roman', 15, 'bold'), bd=2)
passwordEntry.grid(row=2, column=1, padx=40, pady=20)

connectButton=ttk.Button(connectWindow, text='CONECTAR', command=connect)
connectButton.grid(row=3, columnspan=2)

count=0
text=''
def slider():
    global text, count
    if count==len(s):
        count=0
        text=''
    text=text+s[count] #s
    sliderLabel.config(text=text)
    count+=1
    sliderLabel.after(300, slider)

def clock():
    global date, currenttime
    date=time.strftime('%d/%m/%Y')
    currenttime=time.strftime('%H:%M:%S')
    datetimeLabel.config(text=f'Fecha: {date}\nHora: {currenttime}')
    datetimeLabel.after(1000, clock)

#GUI part
root=ttkthemes.ThemedTk()

root.get_themes()

root.get_themes()

root.set_theme('radiance')

root.geometry('1174x680+0+0')
root.resizable(0,0)
root.title('Gestión de Estudiantes')

datetimeLabel=Label(root, font=('times new roman', 18, 'bold'))
datetimeLabel.place(x=5, y=5)
clock()
s='Sistema de Gestion de Estudiantes'
sliderLabel=Label(root, font=('arial', 28, 'italic bold'), width=40)
sliderLabel.place(x=200, y=0)
slider()

connectButton=ttk.Button(root, text='Base de Datos', command=connect_database)
connectButton.place(x=1000, y=10)

```

```

leftFrame=Frame(root)
leftFrame.place(x=50, y=80, width=300, height=600)

logo_image=PhotoImage(file='student.png')
logo_Label=Label(leftFrame, image=logo_image)
logo_Label.grid(row=0, column=0)

addstudentsButton=ttk.Button(leftFrame, text='Agregar Estudiante', width=25,
state=DISABLED, command=lambda :toplevel_data('Agregar Estudiante', 'AGREGAR',
add_data))
addstudentsButton.grid(row=1, column=0, pady=20)

searchstudentsButton=ttk.Button(leftFrame, text='Buscar Estudiante', width=25,
state=DISABLED, command=lambda :toplevel_data('Buscar Estudiante', 'BUSCAR',
search_data))
searchstudentsButton.grid(row=2, column=0, pady=20)

deletestudentsButton=ttk.Button(leftFrame, text='Eliminar Estudiante', width=25,
state=DISABLED, command=delete_student)
deletestudentsButton.grid(row=3, column=0, pady=20)

updatestudentsButton=ttk.Button(leftFrame, text='Actualizar Estudiante', width=25,
state=DISABLED, command=lambda :toplevel_data('Actualizar Estudiante', 'ACTUALIZAR',
update_data))
updatestudentsButton.grid(row=4, column=0, pady=20)

showstudentsButton=ttk.Button(leftFrame, text='Ver Estudiante', width=25,
state=DISABLED, command=show_student)
showstudentsButton.grid(row=5, column=0, pady=20)

exportstudentsButton=ttk.Button(leftFrame, text='Exportar Datos', width=25,
state=DISABLED, command=export_data)
exportstudentsButton.grid(row=6, column=0, pady=20)

exitstudentsButton=ttk.Button(leftFrame, text='SALIR', width=25, command=ixexit)
exitstudentsButton.grid(row=7, column=0, pady=20)

rightFrame=Frame(root)
rightFrame.place(x=350, y=80, width=820, height=600)

scrollBarX=Scrollbar(rightFrame, orient=HORIZONTAL)
scrollBarY=Scrollbar(rightFrame, orient=VERTICAL)

studentTable=ttk.Treeview(rightFrame, columns=('Id', 'Name', 'Mobile', 'Email',
'Address', 'Gender', 'DOB', 'Added Date', 'Added Time'), xscrollcommand=scrollBarX.set,
yscrollcommand=scrollBarY.set)

scrollBarX.config(command=studentTable.xview)
scrollBarY.config(command=studentTable.yview)

scrollBarX.pack(side=BOTTOM, fill=X)
scrollBarY.pack(side=RIGHT, fill=Y)

studentTable.pack(fill=BOTH, expand=1)

studentTable.heading('Id', text='Id')
studentTable.heading('Name', text='Nombre')
studentTable.heading('Mobile', text='Teléfono')
studentTable.heading('Email', text='Email')
studentTable.heading('Address', text='Dirección')
studentTable.heading('Gender', text='Género')
studentTable.heading('DOB', text='Cumpleaños')
studentTable.heading('Added Date', text='Fecha de agregación')
studentTable.heading('Added Time', text='Hora de agregación')

studentTable.column('Id', width=50, anchor=CENTER)
studentTable.column('Name', width=300, anchor=CENTER)
studentTable.column('Email', width=300, anchor=CENTER)
studentTable.column('Mobile', width=200, anchor=CENTER)
studentTable.column('Address', width=300, anchor=CENTER)
studentTable.column('Gender', width=100, anchor=CENTER)
studentTable.column('DOB', width=100, anchor=CENTER)
studentTable.column('Added Date', width=200, anchor=CENTER)
studentTable.column('Added Time', width=200, anchor=CENTER)

style=ttk.Style()

```

```
style.configure('Treeview', rowheight=40, font=('arial', 15, 'bold'),
background='white', fieldbackground='white')
style.configure('Treeview.Hheading', font=('arial', 14, 'bold'), foreground='red')

studentTable.config(show='headings')

root.mainloop()
```

Paso 4: Configuración de Bibliotecas

- Asegúrate de que las bibliotecas necesarias estén instaladas. Abre la terminal integrada en PyCharm y ejecuta los siguientes comandos para instalar las bibliotecas utilizadas en los archivos:

```
pip install pillow
pip install tk
pip install ttkthemes
pip install pandas
pip install pymysql
```

Paso 5: Ejecución de los Archivos

- Para ejecutar cualquiera de los archivos, simplemente haz clic en el archivo en el explorador de proyectos para abrirlo en el editor.
- Luego, busca el botón verde de "Run" en la parte superior derecha del editor y haz clic en él.

¡Listo! Ahora has creado un proyecto "Sistema Estudiantes" en PyCharm y has agregado los archivos login.py y sms.py al mismo. Puedes ejecutar estos archivos individualmente para probar la funcionalidad de tu sistema de gestión de estudiantes. Recuerda que estos pasos son una guía general y pueden variar según tu configuración y versión de PyCharm.



Figura 1. Login de usuario



Figura 2. Conexión a la Base de Datos