Sistemas Expertos
Tema: Basados casos y lógica difusa.

Examen Final

Objetivo:

- Consolidar los conocimientos adquiridos en clase de los sistemas expertos.

Enunciado:

Se desea generar un sistema de recomendación de películas, por tal motivo se va a utiliza una base de datos orientada a grafos y un control de lógica difusa para clasificar el riesgo financiero, el mismo que sera ingresado como atributo del cliente en el sistema recomendador, para lograr esto se describe los pasos a seguir:

1) **Evaluar el riesgo financiero** de sus clientes que requieren la recomendación de películas. Para evaluar el riesgo financiero se toma en cuenta **la edad** del asegurado y su **porcentaje de manejo** durante el año. Para ello se tiene las siguientes reglas y la función de pertinencia. El proceso seguir se describe en el siguiente link: https://medium.com/@javierdiazarca/l%C3%B3gica-difusa-ejercicios-propuestos-b99603ef1bc0.
2) Generar números aleatorios para la edad y el porcentaje de manejo con el objetivo de generar al menos 100 personas y ademas incluir el listado de **películas** vistas y el valor del rating de cada película. Al menos 20 películas y un total de nodos de al menos 250 nodos.
3) Con estos datos aplicar el algoritmo de KNN y Similitud de Coseno para la recomendación de películas, seguir el siguiente tutorial: https://guides.neo4j.com/sandbox/recommendations o https://github.com/MNoorFawi/recommendation-engine-with-neo4j o https://neo4j.com/developer/example-project/.
4) Finalmente realizar alguna interfaz para poder acceder a la recomendación e ingreso de datos y resultados de los procesos.

Generar el Informe en PDF y subir los scripts al repositorio Git para su evaluación.

Fecha de Entrega : 03/08/2021 - 13:55

Criterios de Evaluación
- Sistema lógico difuso: 30%
- Neo4J Knn: 30%
- Informe y resultados: 20%
- GUI. programación y pruebas: 20%

Nota: Subir el sistema en un cuaderno de Python + scripts + PDF.

Examen Final

```
peliculas$ match(n) return n
```

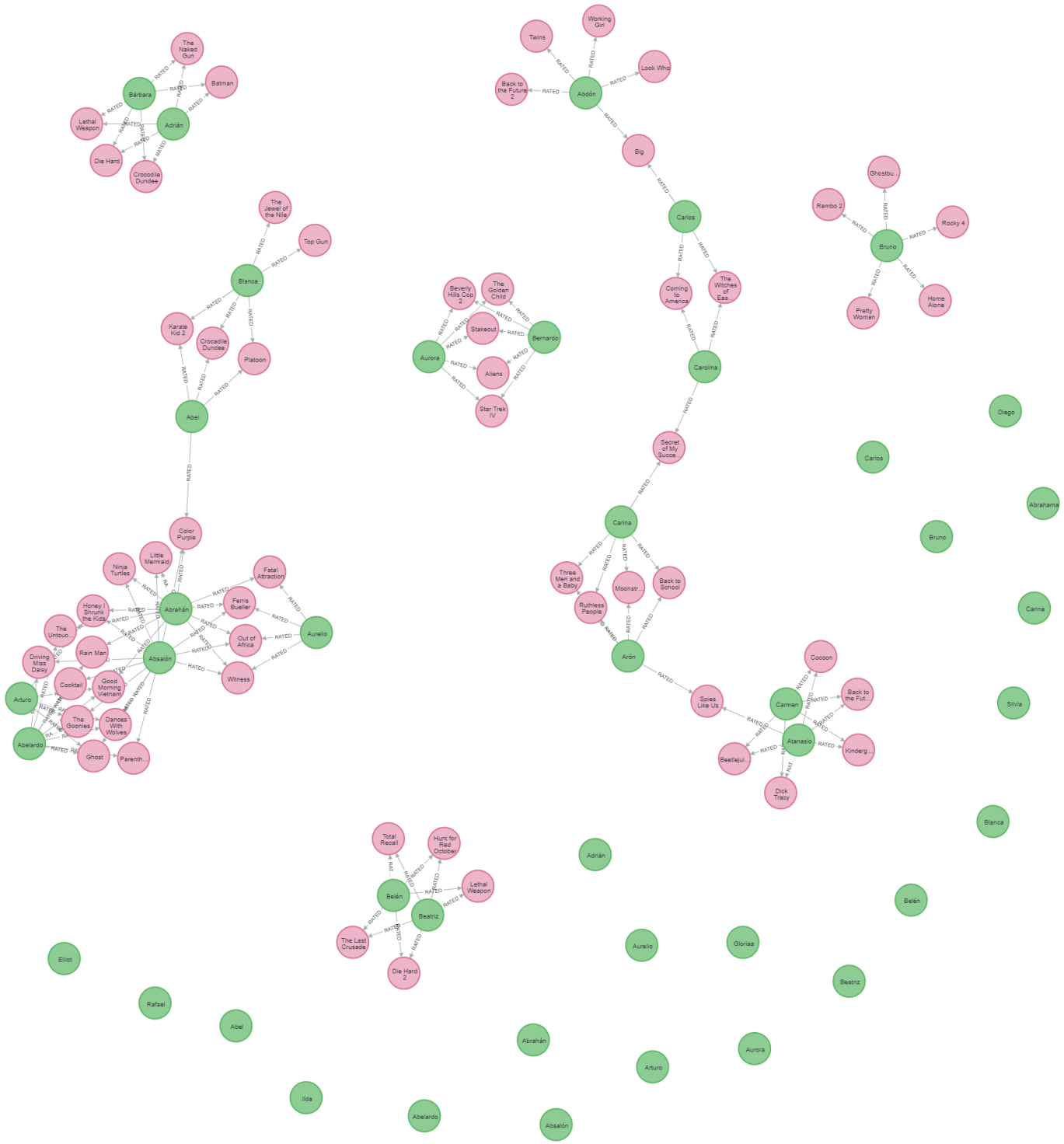
Graph | Table | Text | Code

"n"
{"name":"Rafael","porcentaje":0.55,"edad":25}
{"name":"Elliot","porcentaje":0.78,"edad":33}
{"name":"Ilda","porcentaje":0.76,"edad":43}
{"name":"Abel","porcentaje":0.79,"edad":24}
{"name":"Abelardo","porcentaje":0.05,"edad":44}
{"name":"Abrahán","porcentaje":0.8,"edad":45}
{"name":"Absalón","porcentaje":0.03,"edad":49}
{"name":"Adrián","porcentaje":0.82,"edad":56}
{"name":"Arturo","porcentaje":0.32,"edad":34}
{"name":"Aurelio","porcentaje":0.005,"edad":20}
{"name":"Aurora","porcentaje":0.49,"edad":54}
{"name":"Gloriaa","porcentaje":0.43,"edad":32}
{"name":"Beatriz","porcentaje":0.67,"edad":55}
{"name":"Belén","porcentaje":0.38,"edad":59}

Sistemas Expertos

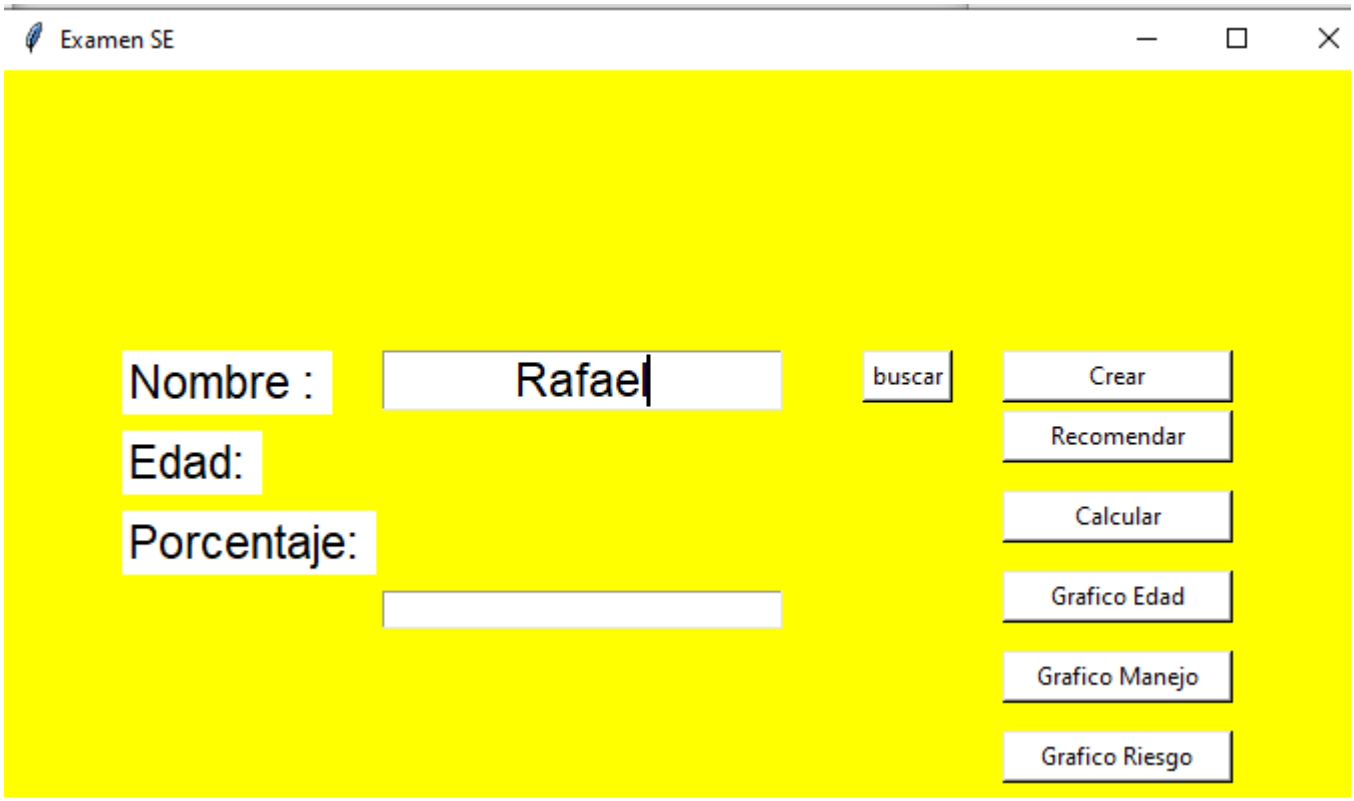
Tema: Basados casos y lógica difusa.

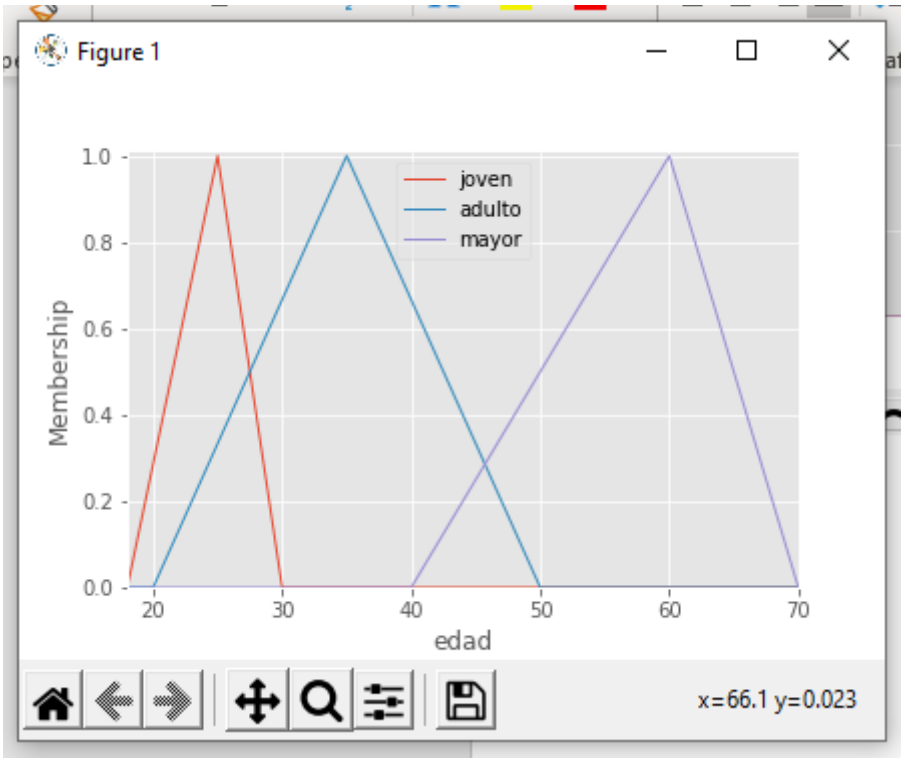
Examen Final



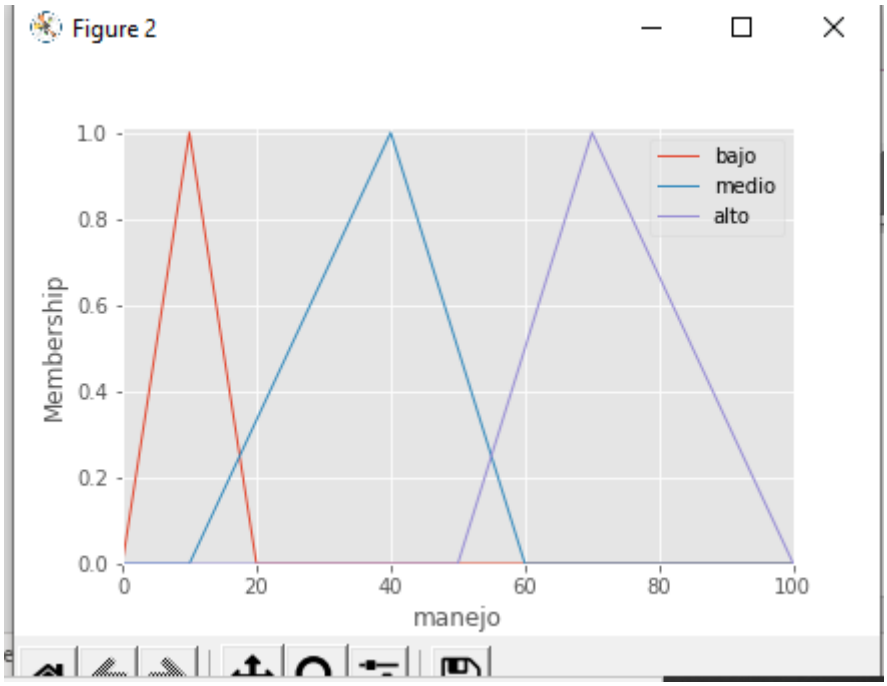
Sistemas Expertos
Tema: Basados casos y lógica difusa.

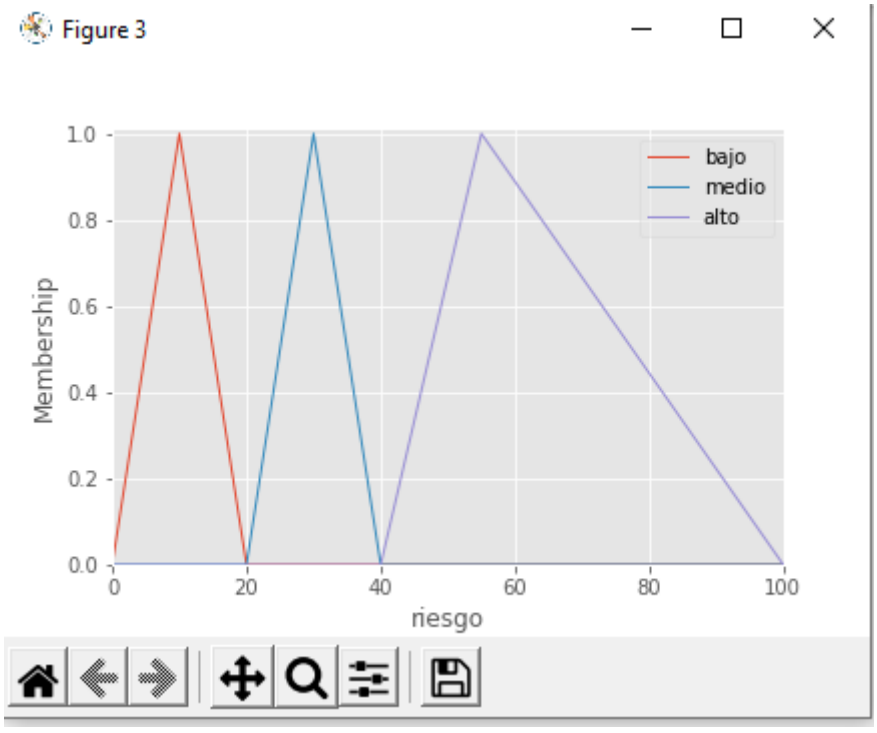
Examen Final





Examen Final





Graficas

In [20]:
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('tkAgg')
import matplotlib.pyplot as plt
from random import randint, uniform,random

def graficas():
    edad.view()
    manejo.view()
    riesgo.view()

def grafica():
    edad.view()

def grafica1():
    manejo.view()

def grafica2():
    riesgo.view()

riesgo = ctrl.Consequent(np.arange(0, 101, 1), 'riesgo')
manejo = ctrl.Antecedent(np.arange(0, 101, 1), 'manejo')
edad = ctrl.Antecedent(np.arange(18, 71, 1), 'edad')

riesgo['bajo'] = fuzz.trimf(riesgo.universe, [0, 10, 20])
riesgo['medio'] = fuzz.trimf(riesgo.universe, [20, 30, 40])
riesgo['alto'] = fuzz.trimf(riesgo.universe, [40, 55, 100])

manejo['bajo'] = fuzz.trimf(manejo.universe, [0, 10, 20])
manejo['medio'] = fuzz.trimf(manejo.universe, [10, 40, 60])
manejo['alto'] = fuzz.trimf(manejo.universe, [50, 70, 100])

edad['joven'] = fuzz.trimf(edad.universe, [18, 25, 30])
edad['adulto'] = fuzz.trimf(edad.universe, [20, 35, 50])
edad['mayor'] = fuzz.trimf(edad.universe, [40, 60, 70])


regla1 = ctrl.Rule(manejo['bajo'] and edad['joven'], riesgo['medio'])
regla2 = ctrl.Rule(manejo['medio'] and edad['joven'], riesgo['alto'])
regla3 = ctrl.Rule(manejo['alto'] and edad['joven'], riesgo['alto'])

regla4 = ctrl.Rule(manejo['bajo'] and edad['adulto'], riesgo['bajo'])
regla5 = ctrl.Rule(manejo['medio'] and edad['adulto'], riesgo['medio'])
regla6 = ctrl.Rule(manejo['alto'] and edad['adulto'], riesgo['alto'])

regla7 = ctrl.Rule(manejo['bajo'] and edad['mayor'], riesgo['medio'])
regla8 = ctrl.Rule(manejo['medio'] and edad['mayor'], riesgo['alto'])
regla9 = ctrl.Rule(manejo['alto'] and edad['mayor'], riesgo['alto'])

regla10 = ctrl.Rule(edad['joven'] and manejo['bajo'], riesgo['medio'])
regla11 = ctrl.Rule(edad['joven'] and manejo['medio'], riesgo['alto'])
regla12 = ctrl.Rule(edad['joven'] and manejo['alto'], riesgo['alto'])
```

```
regla13 = ctrl.Rule(edad['adulto'] and manejo['bajo'], riesgo['bajo'])
regla14 = ctrl.Rule(edad['adulto'] and manejo['medio'], riesgo['medio'])
regla15 = ctrl.Rule(edad['adulto'] and manejo['alto'], riesgo['alto'])

regla16 = ctrl.Rule(edad['mayor'] and manejo['bajo'], riesgo['medio'])
regla17 = ctrl.Rule(edad['mayor'] and manejo['medio'], riesgo['alto'])
regla18 = ctrl.Rule(edad['mayor'] and manejo['alto'], riesgo['alto'])
```

In [15]:

In [27]:
```
from tkinter import *
import tkinter.font as tkFont
from tkinter import ttk
import gym
from neo4j import GraphDatabase
import random


fila=0
columna=0
cuenta=0
nlista=0
porcen=0
edadn=0

def randon():
    porcen=(random.randrange(10, 100))/100
    edadn=random.randrange(10, 80)

ancho=1000
alto=900
my_list1=[]

root1 = Tk()
root1.title("Examen SE")
root1.geometry('1000x500')
root1.configure(bg='white')
fontStyle = tkFont.Font(family="Lucida Grande", size=12)
fontStyle1 = tkFont.Font(family="Lucida Grande", size=17)

fame1= Frame(root1, width=ancho, height=alto, bg='white')
fame1.place(x=10, y=50)
fame1.grid_propagate(False)

frame2= Frame(root1, width=900, height=900, bg='White')
#frame2= Frame(root1, width=ancho-300, height=500, bg='GREEN')

frame2.place(x=50, y=50)
frame2.grid_propagate(False)

class Neo4jService(object):

    def __init__(self, uri, user, password):
        self._driver = GraphDatabase.driver(uri, auth=(user, password))

    def close(self):
        self._driver.close()

    def crear_usuario(self, tx, nombre1, edad1, porcentaje1):
        tx.run("CREATE (cliente:Persona {name:$nombre1, edad:$edad1 ,porcentaje:$
               nombre1=nombre1,edad1=edad1,porcen1=porcentaje1)

    def nombres(self,tx,pelic):
        result = tx.run("MATCH(p:Pelicula) WHERE p.genero= $peli RETURN p.name",
        for line in result:
            listap()
```

```
            mensaje=str(line)
            tam=len(mensaje)-2
            mensajea = mensaje[16:tam]
            my_list1.insert(nlista, mensajea)

    def recomendaciones(self,tx,reco,vjuego):
        result = tx.run("MATCH    (b:Persona)-[r:RATED]->(m:Pelicula), (b)-[s:SIM
                        "WHERE    NOT((a)-[:RATED]->(m))\n"
                        "WITH     m, s.similarity AS similarity, r.rating AS rati
                        "ORDER BY m.name, similarity DESC\n"
                        "WITH     m.name AS pelicula, COLLECT(rating)[0..3] AS ra
                        "WITH     pelicula, REDUCE(s = 0, i IN ratings | s + i)*1
                        "ORDER BY reco DESC\n"
                        "RETURN   pelicula AS Pelicula, reco AS Recommendation",r
        etomo=Label(vjuego,text="TE RECOMENDAMOS :",fg='black', bg='white',font=f
        etomo.place(x=600,y=90)

        frame1 = Frame(vjuego)
        frame1.place(x=600,y=120)


        scrollbar = Scrollbar(frame1)
        scrollbar.pack( side = RIGHT, fill = Y )

        mylist = Listbox(frame1, yscrollcommand = scrollbar.set )
        for line in result:
            mensaje=str(line)
            tam=len(mensaje)-21
            mensajea = mensaje[18:tam]
            mylist.insert(END, mensajea)

        mylist.pack( side = LEFT, fill = BOTH )
        scrollbar.config( command = mylist.yview )

    def datosp(self,tx,frame2,person):
        global edadn
        global porcen
        result=tx.run("MATCH(p:Persona) WHERE p.name=$person RETURN p.edad AS eda
        for line in result:
            r1=(line["edad1"])
            r2=(line["name1"])
            r3=(line["porcentaje1"])
            edadn=int(r1)
            porcen=float(r3)

            usuario2 = Label(frame2, text=str(edadn),fg='black', bg='white', just
            usuario2.place(x=140, y=250, width=200)
            usuario3 = Label(frame2,text=str(porcen), fg='black', bg='white', jus
            usuario3.place(x=140, y=300, width=200)



def crear():
    porcen1=(random.randrange(10, 100))/100
    edadn1=random.randrange(18, 80)
    neo4j = Neo4jService('bolt://localhost:7687', 'peliculas', 'cuenca')
    with neo4j._driver.session() as session:
```

```
        session.write_transaction(neo4j.crear_usuario,usuario1.get(),edadn1,porce

        #esto es para el porcentaje de crear
usuario1 = Entry(frame2, fg='red', bg='white', justify='center')
usuario1.place(x=140, y=210, width=200)

def recomendar():
    neo4j = Neo4jService('bolt://localhost:7687', 'peliculas', 'cuenca')
    with neo4j._driver.session() as session:
        session.write_transaction(neo4j.recomendaciones,usuario1.get(),frame2)

def buscarp():
    neo4j = Neo4jService('bolt://localhost:7687', 'peliculas', 'cuenca')
    with neo4j._driver.session() as session:
        session.write_transaction(neo4j.datosp,frame2,usuario1.get())


def calcular():
    print(edadn,porcen)
    riesgos.input['manejo'] = int(edadn)
    riesgos.input['edad'] = float(porcen)
    riesgos.compute()
    calculado=riesgos.output['riesgo']
    print(calculado)
    txto5 = Label(frame2,text=str(calculado), fg='white', bg='black', font=fontSt
    txto5.place(x=10, y=120)
    riesgo.view(sim=riesgos)

#nombre
txto2 = Label(frame2,text="Nombre : ", fg='black', bg='white', font=fontStyle1)
txto2.place(x=10, y=90)
usuario1 = Entry(frame2, fg='black', bg='white', justify='center',font=fontStyle1
usuario1.place(x=140, y=90, width=200)
btn8 = Button(frame2, text="buscar", bg="white", fg="black", height = 1, width =
btn8.place(x=380, y=90)

txto3 = Label(frame2,text="Edad: ", fg='black', bg='white', font=fontStyle1)
txto3.place(x=10, y=130)

txto4 = Label(frame2,text="Porcentaje: ",fg='black', bg='white', font=fontStyle1)
txto4.place(x=10, y=170)


btn2 = Button(frame2, text="Crear", bg="white", fg="black", height = 1, width = 1
btn2.place(x=450, y=90)

btn3 = Button(frame2, text="Recomendar", bg="white", fg="black", height = 1, widt
btn3.place(x=450, y=120)
btn4 = Button(frame2, text="Calcular", bg="white", fg="black", height = 1, width
btn4.place(x=450, y=160)

btn5 = Button(frame2, text="Grafico Edad", bg="white", fg="black", height = 1, wi
btn5.place(x=450, y=200)

btn6 = Button(frame2, text="Grafico Manejo", bg="white", fg="black", height = 1,
btn6.place(x=450, y=240)
```

```
btn7 = Button(frame2, text="Grafico Riesgo", bg="white", fg="black", height = 1,
btn7.place(x=450, y=280)


def generar(ventana):
    fame1= Frame(root1, width=ancho, height=alto, bg='black')
    fame1.place(x=10, y=50)
    frame1 = Frame(root, background="black")
    fame1.grid_propagate(False)
    return fame1




    neo4j = Neo4jService('bolt://localhost:7687', 'peliculas', 'cuenca')
    with neo4j._driver.session() as session:
        session.read_transaction(neo4j.nombres,generob)


    for i in range(len(my_list1)):
        contador()
        if cuenta == 10:
            fila += 1
            cuenta =0

        panel = Frame(fame1, width=80, height=160, highlightbackground='white', h
        panel.grid(row=fila, column=cuenta, padx=10, pady=3, ipadx=20, ipady=20)

        txto = Label(panel,text=str(my_list1[i]), fg='pink', bg='black', justify=
        txto.place(x=1, y=147, width=115)
        def verp():
            print(txto.cget("text"))

        btn2 = Button(panel, text="ver", bg="white", fg="black", height = 1, widt
        btn2.place(x=20, y=168)


root1.mainloop()
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "c:\users\estsegundorafaelanga\appdata\local\programs\python\python39
\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "C:\Users\ESTSEG~1\AppData\Local\Temp/ipykernel_7536/301217170.py", li
ne 123, in recomendar
    session.write_transaction(neo4j.recomendaciones,usuario1.get(),frame2)
  File "c:\users\estsegundorafaelanga\appdata\local\programs\python\python39
\lib\site-packages\neo4j\work\simple.py", line 434, in write_transaction
    return self._run_transaction(WRITE_ACCESS, transaction_function, *args, *
*kwargs)
  File "c:\users\estsegundorafaelanga\appdata\local\programs\python\python39
\lib\site-packages\neo4j\work\simple.py", line 335, in _run_transaction
    self._open_transaction(access_mode=access_mode, database=self._config.dat
abase, metadata=metadata, timeout=timeout)
  File "c:\users\estsegundorafaelanga\appdata\local\programs\python\python39
```