Simulación
Tema: Inteligencia Artificial 1.

Examen Final

Jonathan Atancuri
Objetivo:

- Consolidar los conocimientos adquiridos en clase sobre la Inteligencia Artificial (IA).

Enunciado:

1. Desarrollar un juego (tema libre) empleando una de las 2 siguientes alternativas:
    1. easyAI
    2. Universe + GYM

El juego deberá implementar algún algoritmo de IA y de igual forma, generar un informe de movimientos, puntajes y quién gana la partida. Se debe tener un juego en donde se tenga un jugar humano y otro utilizando Inteligencia Artificial, finalmente no se puede repetir el juego por más de tres personas por lo que se debe publicar en el foro el juego seleccionado.

2. Dentro del juego el usuario puede registrar he ingresar los gustos de alguna área basadas en el lugar geográfico por ejemplo: comida, películas, lugares turísticos etc.

3. En base a la información proporcionada se deberá generar un sistema que permita mostrar lugares de interés, para ello tomar los datos de las tareas y pruebas dentro de una base de datos orientadas a grafos.

4. Realizar el sistema con una interfaz gráfica y almacenar los puntajes y datos de los usuarios o jugadores.

Código y documentos de entrega: Se deberá entregar un informe con el procesos dentro del mismo tener capturas del uso del juego y generar un documento en PDF de validación y pruebas. Finalmente subir todo al repositorio incluido los códigos fuentes

Criterios de Evaluación:

- Neo4J y Búsquedas : 30%
- Juego IA: 30%
- GUI: 20%
- Informe PDF: 20%
- Usabilidad: 10%

Fecha de entrega: **07/02/2021** – 23**:55.**

Nota: Cualquier pregunta o duda con respecto al examen escribirme por correo electrónico o whatssapp.

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

Máncala

Se llama mancala a una amplia familia de juegos de tablero fundamentalmente africanos y también asiáticos que comparten una serie de características comunes: el tablero con receptáculos u hoyos, las 'semillas' o fichas y el juego que se denomina siembra. La denominación kalaha puede provenir de su frecuente uso en las poblaciones del desierto kalahari.

Los componentes del juego son:

El tablero está compuesto por una serie de receptáculos (que pueden ser agujeros hechos en el suelo o en un bloque de madera) organizados en filas.

Estos receptáculos contienen una serie de piezas -48- (semillas, guijarros, canicas) indiferenciadas que son los elementos móviles del juego.

El movimiento básico es el que habitualmente se llama "de siembra", que consiste en tomar todas las piezas contenidas en uno de los receptáculos e irlas depositando de una en una en receptáculos consecutivos a partir del siguiente al que las contenía.

Tipos de Mancalas

Manqala de dos filas (ejemplo típico el oware, wari o agualé):

en estos juegos cada una de las filas pertenece a un jugador, el movimiento de siembra pasa las semillas de una fila a otra cuando corresponde, sólo se pueden tomar semillas de un receptáculo de la fila propia para mover, se captura (de diversas maneras, según el juego en concreto), y cuando se da la condición de final de partida gana el que haya capturado al menos 25 semillas.

Manqala de cuatro filas (ejemplo típico el bao, bawo o omweso):

en estos juegos a cada jugador pertenecen tres filas consecutivas. El movimiento de siembra mantiene las semillas en las dos filas del jugador que mueve, pero en determinadas circunstancias pueden tomarse las semillas de un receptáculo del rival y unirlas a las de un receptáculo propio. Cuando se da la condición de final de partida gana el que tenga más semillas en sus dos filas.

Manqala de número impar de filas:

Aquí entran las versiones tanto de una fila (solitarios) como de tres.

El juego se trata en tener mayor puntaje gana, cada jugador tiene 6 casillas con 4 semillas cada jugador , este juego se juega en sentido de las manecillas del reloj.

Simulación
Tema: Inteligencia Artificial 1.

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

Examen Final

```
MANCALA - jugado desde tiempos remotos

Comienzas eres el jugador player 1
Tus casillas son a, b, c, d, e, f

Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
04 04 04 04 04 04
04 04 04 04 04 04
a  b  c  d  e  f

Player 1 what do you play ? c

Move #1: player 1 plays c :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
04 04 04 04 04 05
04 04 00 05 05 05
a  b  c  d  e  f
```

Vamos a jugar un poco

El juego que se implemento es mancala es juego de Sudáfrica Que se necesita de un poco de lógica para poder jugarla, Constade dos jugadores. El player 1 es el jugador humano y el player 2 en la maquina de inteligencia artificial , cada que avanza las partidas van acumulando puntos los jugadores , este puede quedar empa te o ganar , siempre y cuando tengan muchas semillas acumuladas.

```
MANCALA - jugado desde tiempos remotos

Comienzas eres el jugador player 1
Tus casillas son a, b, c, d, e, f

Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
04 04 04 04 04 04
04 04 04 04 04 04
a  b  c  d  e  f

Player 1 what do you play ? a

Player 1 what do you play ? b

Player 1 what do you play ? c

Move #1: player 1 plays c :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
04 04 04 04 04 05
04 04 00 05 05 05
a  b  c  d  e  f

Move #2: player 2 plays i :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
05 05 05 00 04 05
05 04 00 05 05 05
a  b  c  d  e  f

Move #3: player 1 plays f :
Puntuje: Player1 0 /  Maquina 0
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
l  k  j  i  h  g
05 06 06 01 05 06
05 04 00 05 05 00
a  b  c  d  e  f

Move #4: player 2 plays g :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
06 07 07 02 06 00
06 04 00 05 05 00
a  b  c  d  e  f

Player 1 what do you play ? a

Move #5: player 1 plays a :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
06 07 07 02 06 01
00 05 01 06 06 01
a  b  c  d  e  f

Move #6: player 2 plays h :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
07 08 08 03 00 01
01 06 01 06 06 01
a  b  c  d  e  f

Player 1 what do you play ? b

Move #7: player 1 plays b :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
07 08 08 03 01 02
01 00 02 07 07 02
a  b  c  d  e  f

Move #8: player 2 plays j :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
08 09 00 03 01 02
02 01 03 08 08 03
a  b  c  d  e  f

Player 1 what do you play ? d

Move #9: player 1 plays d :
Puntuje: Player1 0 /  Maquina 0
l  k  j  i  h  g
09 10 01 04 02 03
02 01 03 00 09 04
a  b  c  d  e  f

Move #10: player 2 plays i :

Player 1 what do you play ? f
```

Simulación
Tema: Inteligencia Artificial 1.

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

Examen Final

```
Move #11: player 1 plays f :
Puntuje: Player1 0 /  Maquina 3
l  k  j  i  h  g
10 11 03 01 03 04
00 01 03 00 09 00
a  b  c  d  e  f

Move #12: player 2 plays k :
Puntuje: Player1 0 /  Maquina 3
l  k  j  i  h  g
11 00 04 02 04 05
01 02 04 01 10 01
a  b  c  d  e  f

Player 1 what do you play ? c

Move #13: player 1 plays c :
Puntuje: Player1 0 /  Maquina 3
l  k  j  i  h  g
11 00 04 02 04 06
01 02 00 02 11 02
a  b  c  d  e  f

Move #14: player 2 plays g :
Puntuje: Player1 0 /  Maquina 5
l  k  j  i  h  g
12 01 05 03 05 00
00 02 00 02 11 02
a  b  c  d  e  f

Player 1 what do you play ? f

Move #15: player 1 plays f :
Puntuje: Player1 0 /  Maquina 5
l  k  j  i  h  g
12 01 05 03 06 01
00 02 00 02 11 00
a  b  c  d  e  f

Move #16: player 2 plays j :
Puntuje: Player1 0 /  Maquina 5
l  k  j  i  h  g
13 02 00 03 06 01
01 03 01 02 11 00
a  b  c  d  e  f

Player 1 what do you play ? a

Player 1 what do you play ? b

Player 1 what do you play ? c

Player 1 what do you play ? d

Player 1 what do you play ? e
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
Move #17: player 1 plays e :
Puntuje: Player1 0 /  Maquina 5
l  k  j  i  h  g
14 03 01 04 07 02
02 04 02 03 00 01
a  b  c  d  e  f

Move #18: player 2 plays i :
Puntuje: Player1 0 /  Maquina 8
l  k  j  i  h  g
15 04 02 00 07 02
00 04 02 03 00 01
a  b  c  d  e  f

Player 1 what do you play ? d

Move #19: player 1 plays d :
Puntuje: Player1 0 /  Maquina 8
l  k  j  i  h  g
15 04 02 00 07 03
00 04 02 00 01 02
a  b  c  d  e  f

Move #20: player 2 plays h :
Puntuje: Player1 0 /  Maquina 8
l  k  j  i  h  g
16 05 03 01 00 03
01 05 03 00 01 02
a  b  c  d  e  f

Player 1 what do you play ? b

Move #21: player 1 plays b :
Puntuje: Player1 0 /  Maquina 8
l  k  j  i  h  g
16 05 03 01 00 04
01 00 04 01 02 03
a  b  c  d  e  f

Move #22: player 2 plays j :
Puntuje: Player1 0 /  Maquina 10
l  k  j  i  h  g
17 06 00 01 00 04
00 00 04 01 02 03
a  b  c  d  e  f

Player 1 what do you play ? c

Move #23: player 1 plays c :
Puntuje: Player1 0 /  Maquina 10
l  k  j  i  h  g
17 06 00 01 00 05
00 00 00 02 03 04
a  b  c  d  e  f
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
Move #24: player 2 plays k :
Puntuje: Player1 0 /  Maquina 10
l  k  j  i  h  g
18 00 00 01 00 05
01 01 01 03 04 04
a  b  c  d  e  f

Player 1 what do you play ? e

Move #25: player 1 plays e :
Puntuje: Player1 0 /  Maquina 10
l  k  j  i  h  g
18 00 00 02 01 06
01 01 01 03 00 05
a  b  c  d  e  f

Move #26: player 2 plays g :
Puntuje: Player1 0 /  Maquina 12
l  k  j  i  h  g
19 01 01 03 02 00
00 01 01 03 00 05
a  b  c  d  e  f

Player 1 what do you play ? f

Move #27: player 1 plays f :
Puntuje: Player1 0 /  Maquina 12
l  k  j  i  h  g
19 02 02 04 03 01
00 01 01 03 00 00
a  b  c  d  e  f

Move #28: player 2 plays i :
Puntuje: Player1 0 /  Maquina 12
l  k  j  i  h  g
20 03 03 00 03 01
01 01 01 03 00 00
a  b  c  d  e  f

Player 1 what do you play ? d

Move #29: player 1 plays d :
Puntuje: Player1 0 /  Maquina 12
l  k  j  i  h  g
20 03 03 00 03 02
01 01 01 00 01 01
a  b  c  d  e  f

Move #30: player 2 plays j :
Puntuje: Player1 0 /  Maquina 14
l  k  j  i  h  g
21 04 00 00 03 02
00 01 01 00 01 01
a  b  c  d  e  f

Player 1 what do you play ? a
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
Player 1 what do you play ? b

Player 1 what do you play ? c

Player 1 what do you play ? e

Player 1 what do you play ? f

Move #31: player 1 plays f :
Puntuje: Player1 0 /  Maquina 14
l  k  j  i  h  g
21 04 00 00 03 03
00 01 01 00 01 00
a  b  c  d  e  f

Move #32: player 2 plays l :
Puntuje: Player1 0 /  Maquina 14
l  k  j  i  h  g
00 05 02 02 05 05
02 03 03 02 03 02
a  b  c  d  e  f

Player 1 what do you play ? f

Move #33: player 1 plays f :
Puntuje: Player1 0 /  Maquina 14
l  k  j  i  h  g
00 05 02 02 06 06
02 03 03 02 03 00
a  b  c  d  e  f

Move #34: player 2 plays g :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
01 06 03 03 07 00
00 03 03 02 03 00
a  b  c  d  e  f

Player 1 what do you play ? b

Player 1 what do you play ? c

Player 1 what do you play ? e

Move #35: player 1 plays e :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
01 06 03 03 08 01
00 03 03 02 00 01
a  b  c  d  e  f

Move #36: player 2 plays h :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
02 07 04 04 00 01
```

Simulación
Tema: Inteligencia Artificial 1.

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

Examen Final

```
01 04 04 03 00 01
a  b  c  d  e  f

Player 1 what do you play ? d

Move #37: player 1 plays d :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
02 07 04 04 00 02
01 04 04 00 01 02
a  b  c  d  e  f

Move #38: player 2 plays j :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
03 08 00 04 00 02
02 05 04 00 01 02
a  b  c  d  e  f

Player 1 what do you play ? c

Move #39: player 1 plays c :
Puntuje: Player1 0 /  Maquina 17
l  k  j  i  h  g
03 08 00 04 00 03
02 05 00 01 02 03
a  b  c  d  e  f

Move #40: player 2 plays i :
Puntuje: Player1 0 /  Maquina 20
l  k  j  i  h  g
04 09 01 00 00 03
00 05 00 01 02 03
a  b  c  d  e  f

Player 1 what do you play ? b

Move #41: player 1 plays b :
Puntuje: Player1 0 /  Maquina 20
l  k  j  i  h  g
04 09 01 00 00 04
00 00 01 02 03 04
a  b  c  d  e  f

Move #42: player 2 plays k :
Puntuje: Player1 0 /  Maquina 20
l  k  j  i  h  g
05 00 01 00 01 05
01 01 02 03 04 05
a  b  c  d  e  f


Move #43: player 1 plays d :
Puntuje: Player1 0 /  Maquina 20
l  k  j  i  h  g
05 00 01 00 01 06
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
01 01 02 00 05 06
a  b  c  d  e  f

Move #44: player 2 plays g :
Puntuje: Player1 0 /  Maquina 22
l  k  j  i  h  g
06 01 02 01 02 00
00 01 02 00 05 06
a  b  c  d  e  f

Player 1 what do you play ? e

Move #45: player 1 plays e :
Puntuje: Player1 0 /  Maquina 22
l  k  j  i  h  g
06 01 03 02 03 01
00 01 02 00 00 07
a  b  c  d  e  f

Move #46: player 2 plays l :
Puntuje: Player1 0 /  Maquina 22
l  k  j  i  h  g
00 01 03 02 03 01
01 02 03 01 01 08
a  b  c  d  e  f

Player 1 what do you play ? f

Move #47: player 1 plays f :
Puntuje: Player1 0 /  Maquina 22
l  k  j  i  h  g
01 02 04 03 04 02
02 03 03 01 01 00
a  b  c  d  e  f

Move #48: player 2 plays k :
Puntuje: Player1 0 /  Maquina 25
l  k  j  i  h  g
02 00 04 03 04 02
00 03 03 01 01 00
a  b  c  d  e  f
PERDISTE .
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
        print( Parece que tenemos un empate. )

Player 1 what do you play ? d

Player 1 what do you play ? e

Player 1 what do you play ? f

Move #47: player 1 plays f :
Puntuje: Player1 0 /  Maquina 22
l  k  j  i  h  g
01 02 04 03 04 02
02 03 03 01 01 00
a  b  c  d  e  f

Move #48: player 2 plays k :
Puntuje: Player1 0 /  Maquina 25
l  k  j  i  h  g
02 00 04 03 04 02
00 03 03 01 01 00
a  b  c  d  e  f
PERDISTE .
```

Codigo del juego :

Simulación
Tema: Inteligencia Artificial 1.

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

Examen Final

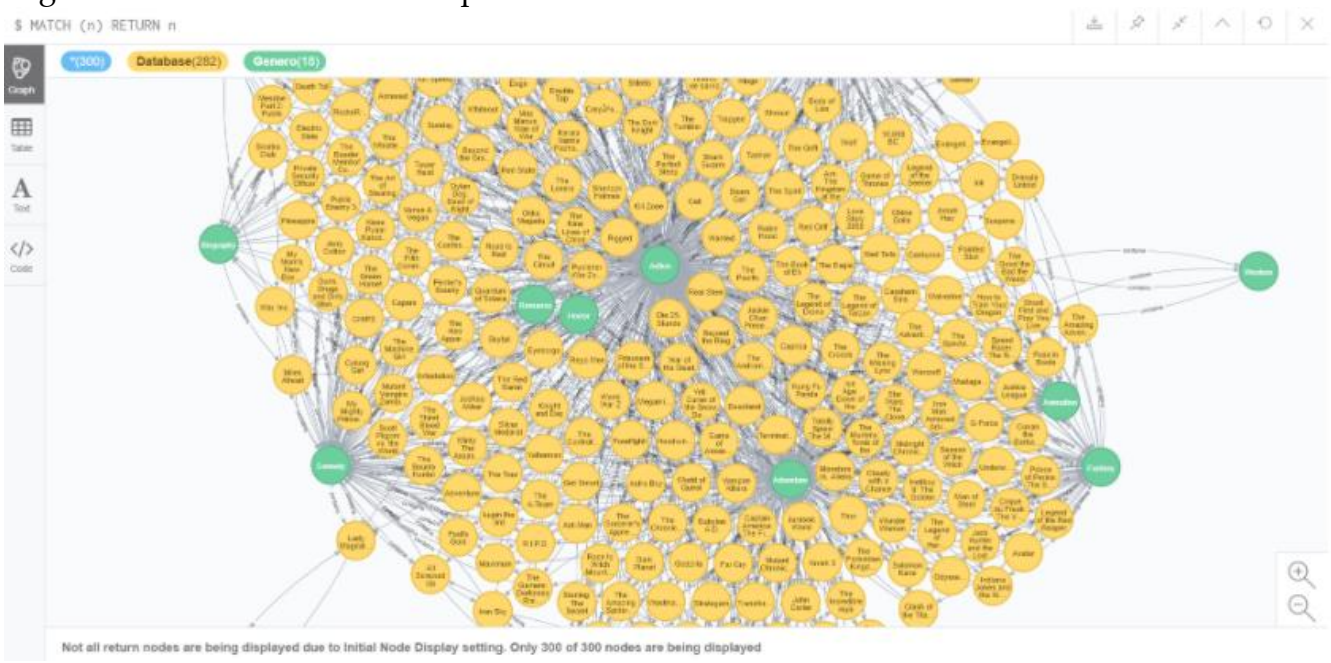
```
try:
    import numpy as np
except ImportError:
    print("Lo sentimos... Necesita Instalar numpy para que funcione el juego.")
    print("MANCALA - jugado desde tiempos remotos")
    raise

from easyAI import TwoPlayersGame


class Awele(TwoPlayersGame):

    print()
    print("MANCALA - jugado desde tiempos remotos")
    print()
    print("Comienzas eres el jugador player 1")
    print("Tus casillas son a, b, c, d, e, f")
    print()
    def __init__(self, players):
        for i, player in enumerate(players):
            player.score = 0
            player.isstarved = False
            player.camp = i
        self.players = players

        # Initial configuration of the board.
        # holes are indexed by a,b,c,d...
        self.board = [4, 4, 4, 4, 4, 4,
                      4, 4, 4, 4, 4, 4]

        self.nplayer = 1  # player 1 starts.

    def make_move(self, move):
        if move == "None":
            self.player.isstarved = True
            s = 6 * self.opponent.camp
            self.player.score += sum(self.board[s:s + 6])
            return

        move = 'abcdefghijkl'.index(move)

        pos = move
        for i in range(self.board[move]):  #DEAL
            pos = (pos + 1) % 12
            if pos == move:
                pos = (pos + 1) % 12
            self.board[pos] += 1
        self.board[move] = 0

        while ((pos / 12) == self.opponent.camp
               and (self.board[pos] in [2, 3])):  # TAKE
            self.player.score += self.board[pos]
            self.board[pos] = 0
            pos = (pos - 1) % 12
    def possible_moves(self):
```

Simulación
Tema: Inteligencia Artificial 1.

Examen Final

```
        if self.nplayer == 1:
            if max(self.board[:6]) == 0: return ['None']
            moves = [i for i in range(6) if (self.board[i] >= 6 - i)]
            if moves == []:
                moves = [i for i in range(6) if self.board[i] != 0]
        else:
            if max(self.board[6:]) == 0: return ['None']
            moves = [i for i in range(6,12) if (self.board[i] >= 12-i)]
            if moves == []:
                moves = [i for i in range(6, 12) if self.board[i] != 0]

        return ['abcdefghijkl'[u] for u in moves]

    def show(self):
        print("Puntuje: Player1 %d /  Maquina %d" % tuple(p.score for p in self.players))
        print('  '.join('lkjihg'))
        print(' '.join(["%02d" % i for i in self.board[-1:-7:-1]]))
        print(' '.join(["%02d" % i for i in self.board[:6]]))
        print('  '.join('abcdef'))

    def lose(self):
        return self.opponent.score > 24

    def is_over(self):
        return ( self.lose() or
                 sum(self.board) < 7 or
                 self.opponent.isstarved )


if __name__ == "__main__":
    from easyAI import Human_Player, AI_Player, Negamax

    scoring = lambda game: game.player.score - game.opponent.score
    ai = Negamax(6, scoring)
    game = Awele([Human_Player(), AI_Player(ai)])

    game.play()

    if game.player.score > game.opponent.score:
        print("GANASTE .")
    elif game.player.score < game.opponent.score:
        print("PERDISTE .")
    else:
        print("Parece que tenemos un empate.")
```

Algorimo de recomendación de películas



Examen Final

def add_Excel():

Este método se encarga de agregar una base de datos de Excel en el grafo. Dentro de esta base de datos una columna está reservada para el nombre de película o series y otras tres columnas para 3 géneros que posee cada película o serie.

```
def add_Excel():
    lista_gen = []
    for x in range(sheet.nrows):
        name = sheet.cell_value(x,0)
        gen1 = sheet.cell_value(x,1)
        gen2 = sheet.cell_value(x,2)
        gen3 = sheet.cell_value(x,3)

        lista_gen = []

        lista_gen.append(gen1)
        lista_gen.append(gen2)
        lista_gen.append(gen3)

        lista_gen.sort()

        gen1 = lista_gen[0]
        gen2 = lista_gen[1]
        gen3 = lista_gen[2]

        generos = []

        generos.append(gen1)
        generos.append(gen2)
        generos.append(gen3)

        database[name] = generos

        unidad = db.nodes.create(nombre=name, genero1=gen1, genero2=gen2, genero3=gen3)
        data8.add(unidad)
```

Luego de agregar los nodos se realizarán las relaciones hacia los géneros mostrados en el resto del código de def add_Excel mostrados a continuación. Se intenta realizar la relación entre la película y gen1, gen2 y gen3. En el caso de que no se pueda (porque el nodo todavía no existe) se crea el género en el label Genero y se intenta hacer de nuevo la relación.

```
    try:
        unidad.relationships.create("contains", gen.get(genero=gen1)[0])
        gen.get(genero=gen1)[0].relationships.create("contains", unidad)
    except Exception:
        genNode = db.nodes.create(genero=gen1)
        gen.add(genNode)
        unidad.relationships.create("contains", gen.get(genero=gen1)[0])
        gen.get(genero=gen1)[0].relationships.create("contains", unidad)

    try:
        unidad.relationships.create("contains", gen.get(genero=gen2)[0])
        gen.get(genero=gen2)[0].relationships.create("contains", unidad)
    except Exception:
        genNode = db.nodes.create(genero=gen2)
        gen.add(genNode)
        unidad.relationships.create("contains", gen.get(genero=gen2)[0])
        gen.get(genero=gen2)[0].relationships.create("contains", unidad)

    try:
        unidad.relationships.create("contains", gen.get(genero=gen3)[0])
        gen.get(genero=gen3)[0].relationships.create("contains", unidad)
    except Exception:
        genNode = db.nodes.create(genero=gen3)
        gen.add(genNode)
        unidad.relationships.create("contains", gen.get(genero=gen3)[0])
        gen.get(genero=gen3)[0].relationships.create("contains", unidad)
```

add_Database():

Este método se encarga de ingresar elementos independientes al grafo en caso de que no haya una película o serie en la base de datos. Además de ingresar el nodo se realizarán las relaciones con los géneros. Al igual que en add_Excel(), en caso de que no exista un género, se crearán los géneros necesarios en el label Genero para poder realizar la relación.

def watch():

Este método no solo se encarga de hacer que los géneros de la película o serie se agreguen en la

lista para la recomendación, también se encarga de dar el parámetro para el método de popular_topics() para mostrar al usuario cuales son los 5 géneros que más ha visto y realizar la recomendación.

```
def watch():
    name = raw_input("Insert name: ")

    try:
        query = "MATCH (n:Database) WHERE n.nombre='"+name+"' RETURN n.genero1, n.genero2, n.genero3"
        results = db.query(query, data_contents=True)
        a = results.rows
        for x in a:
            historial.append(x[0])
            historial.append(x[1])
            historial.append(x[2])



    except Exception:
        print("The movie or TV show you were looking for is not in the database, you can add it by going to option 1")

    popular_topics(name)
```

def popular_topics(name):

Este método se encarga de mostra los 5 métodos más vistos y recomendar películas y series, la recomendación empezará desde el nodo que fue ingresado en el método watch(). La manera para determinar cuáles son los géneros que más se repiten es recorriendo toda la lista utilizando un ciclo for donde en caso de que no se encuentra la palabra en el diccionario, agregarla como llave y darle un valor de 1, en caso de que si se encuentre la palabra como llave, sumar 1 al valor que posee.

Luego se ordena el diccionario utilizando sorted(word_counter, key = word_counter.get, reverse = True) y se elijen las primeras 5 categorias (las 5 categorías que se repiten más veces).

```
def popular_topics(name):
    nombre = name
    #diccionario que determinarÃ¡ cuales son los 5 generos mÃ¡s vistos
    top_5 = []
    #por cada genero en la lista....
    word_counter = {}
    for word in historial:
        if word in word_counter:
            word_counter[word] += 1
        else:
            word_counter[word] = 1

    popular_words = sorted(word_counter, key = word_counter.get, reverse = True)

    top_5 = popular_words[:5]

    #se ordenan los generos en orden alfabetico
    lista = []
    print ("Most watched genres: ")
    for x in top_5:
        lista.append(x)
        print (x)
```

Por último, para poder recomendar las películas y series, se tendrá que utilizar el código (manera que debe escribirse en neo4j Desktop) match (n:Database{nombre:'"+nombre+"'})-[:contains*1..3]->(a:Database{generoX:'"+top_5[generos principales]+"'}) return collect(distinct

a.nombre).

donde nombre es la película o serie que vió el usuario (desde donde se empezará a buscar), generoX es si se quiere buscar en generos1, genero2 o genero3 y top_5[] tendrá la posición cero (genero que más se ve) , uno (2do) o dos (3ero).

Esta búsqueda se realizara para los tres género que más se repiten y la búsqueda para los tres será en generos1, generos2 y generos3.

```
def show_genre():
    genre = raw_input("Insert genre: ")
    try:
        query = "MATCH (n:Database {genero1:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print (x)

    except Exception:
        pass

    try:
        query = "MATCH (n:Database {genero2:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print (x)

    except Exception:
        pass

    try:
        query = "MATCH (n:Database {genero3:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print (x)

    except Exception:
        pass
```

Conclusión: En conclusión se puede decir que esta librería de EasyAI permite que un agente inteligente pueda pueda interactuar con un usuario que seria un humano , es muy importante destacar que nosotros podemos definir el nivel de dificultad de nuestro juego y que hoy en día la inteligencias artificial es muy sorprendente ya que no necesita de una persona humana para jugar , ya que este agente inteligente se encarga de todo y muy útil la librería para programar juegos.

Link repositorio:
https://github.com/JonathanAtancuri3218/prueba/tree/master/2do%20Interciclo/ExamenFinal-IA