

Inteligencia Artificial

Examen

Nombre: Andrés Zeas G.

- Importamos las siguientes librerías y el driver para conectar a la base de datos.

```
import logging
from neo4j import GraphDatabase
from neo4j.exceptions import ServiceUnavailable
from facebook_scraper import get_posts
class Neo4jService:

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

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

- Definimos funciones para crear los nodos.

```
#CREACION DE NODOS

def crear_Alcaldes(self, tx, nombre):
    tx.run("CREATE (:Alcaldes {nombre: $nombre})", nombre=nombre)

def crear_alcalde(self, tx, nombre):
    tx.run("CREATE (:Alcalde {nombre: $nombre})", nombre=nombre)

def crear_publicacion(self, tx, nombre):
    tx.run("CREATE (:Publicacion {nombre: $nombre})", nombre=nombre)

def crear_fecha(self, tx, nombre):
    tx.run("CREATE (:Fecha {nombre: $nombre})", nombre=nombre)

def crear_texto(self, tx, nombre):
    tx.run("CREATE (:Texto {nombre: $nombre})", nombre=nombre)

def crear_likes(self, tx, nombre):
    tx.run("CREATE (:Likes {nombre: $nombre})", nombre=nombre)

def crear_comentarios(self, tx, nombre):
    tx.run("CREATE (:Comentarios {nombre: $nombre})", nombre=nombre)
```

- Creamos las relaciones

#CREACION DE RELACIONES

```
def crear_relacion_Alcs_Alc(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Alcaldes {nombre: $nombre1}) "
           "MATCH (b:Alcalde {nombre: $nombre2}) "
           "MERGE (a)-[:Alcaldess_Alcalde]->(b)",
           nombre1=nombre1, nombre2=nombre2)

def crear_relacion_alc_publicacion(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Alcalde {nombre: $nombre1}) "
           "MATCH (b:Publicacion {nombre: $nombre2}) "
           "MERGE (a)-[:Alcalde_Publicacion]->(b)",
           nombre1=nombre1, nombre2=nombre2)

def crear_relacion_public_fecha(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
           "MATCH (b:Fecha {nombre: $nombre2}) "
           "MERGE (b)-[:Publicacion_Fecha]->(a)",
           nombre1=nombre1, nombre2=nombre2)

def crear_relacion_public_texto(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
           "MATCH (b:Texto {nombre: $nombre2}) "
           "MERGE (b)-[:Publicacion_Texto]->(a)",
           nombre1=nombre1, nombre2=nombre2)

def crear_relacion_public_like(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
           "MATCH (b:Likes {nombre: $nombre2}) "
           "MERGE (b)-[:Publicacion_Like]->(a)",
           nombre1=nombre1, nombre2=nombre2)

def crear_relacion_public_comen(self, tx, nombre1, nombre2):
    tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
           "MATCH (b:Comentarios {nombre: $nombre2}) "
           "MERGE (b)-[:Publicacion_Comentarios]->(a)",
           nombre1=nombre1, nombre2=nombre2)
```

- Conectamos la base de datos Neo4j con el usuario y contraseña definido.

#CONEXION A LA BASE DE DATOS

#Y WEB SCRAPING

```
neo4j = Neo4jService('bolt://localhost:7687', 'neo4j', 'admin')
with neo4j._driver.session() as session:
```

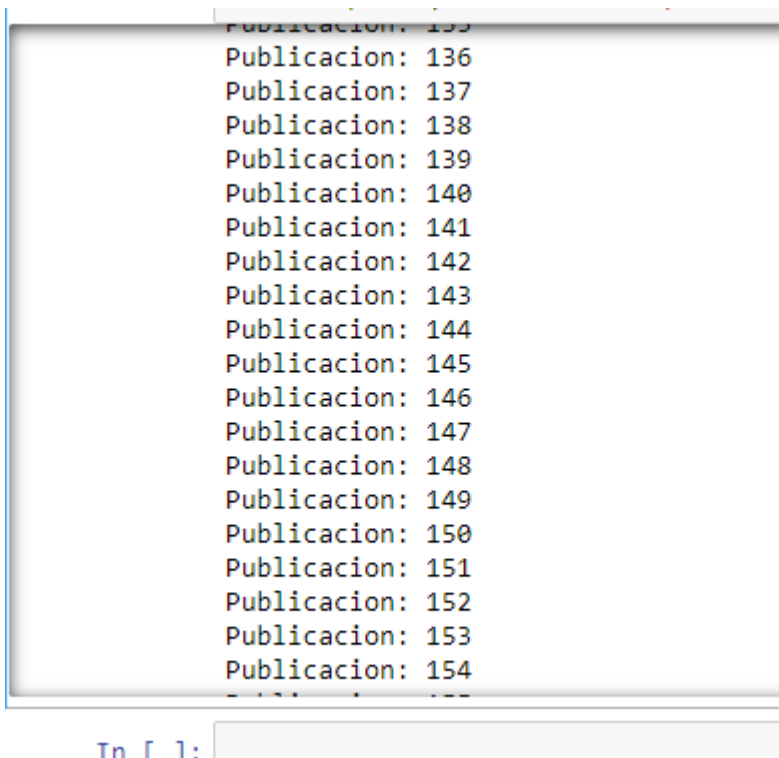
- Una vez iniciada la sesión hacemos uso de las herramientas de scraping de Facebook para extraer las publicaciones junto con la fecha, comentarios y likes de estas.

```

with neo4j._driver.session() as session:
    contador = -1
    session.write_transaction(neo4j.crear_Alcaldes, "Alcaldes")
    session.write_transaction(neo4j.crear_alcalde, "Pedro Palacios")
    session.write_transaction(neo4j.crear_relacion_Alcs_Alc, "Alcaldes", "Pedro Palacios")
    for post in get_posts('PedroPalaciosU', pages=52, timeout=1):
        contador = contador + 1
        publicacion = "Publicacion: " + str(contador)
        fecha = "Fecha de la publicacion N° " + str(contador) + ": " + str(post['time'])
        text = "Contenido de la publicacion N° " + str(contador) + ": " + str(post['text'])
        likes = "Likes de la publicacion N° " + str(contador) + ": " + str(post['likes'])
        comentarios = "Numero de comentarios de la publicacion N° " + str(contador) + ": " + str(post['comments'])
        session.write_transaction(neo4j.crear_publicacion, publicacion)
        session.write_transaction(neo4j.crear_fecha, fecha)
        session.write_transaction(neo4j.crear_texto, text)
        session.write_transaction(neo4j.crear_likes, likes)
        session.write_transaction(neo4j.crear_comentarios, comentarios)
        session.write_transaction(neo4j.crear_relacion_alc_publicacion, "Pedro Palacios", publicacion)
        session.write_transaction(neo4j.crear_relacion_public_fecha, publicacion, fecha)
        session.write_transaction(neo4j.crear_relacion_public_texto, publicacion, text)
        session.write_transaction(neo4j.crear_relacion_public_like, publicacion, likes)
        session.write_transaction(neo4j.crear_relacion_public_comen, publicacion, comentarios)
    print(publicacion)
print("Fin de la busqueda...")

```

- En pantalla se muestran las publicaciones que el sistema va encontrado.



```

Publicacion: 135
Publicacion: 136
Publicacion: 137
Publicacion: 138
Publicacion: 139
Publicacion: 140
Publicacion: 141
Publicacion: 142
Publicacion: 143
Publicacion: 144
Publicacion: 145
Publicacion: 146
Publicacion: 147
Publicacion: 148
Publicacion: 149
Publicacion: 150
Publicacion: 151
Publicacion: 152
Publicacion: 153
Publicacion: 154

```

In []:

- Y en la base de datos podemos observar todos los datos que se han extraído.

