Universidad Politécnica Salesiana

<u>Inteligencia Artificial</u>

Tema:

Examen de Interciclo

Integrantes:

Carlos Álvarez

Docente:

Ing. Diego Quisi

- **Análisis**
  - **Librerías necesarias**

```
import logging
from neo4j import GraphDatabase
from neo4j.exceptions import ServiceUnavailable
from facebook_scraper import get_posts
```

  - **Funciones para crear los nodos**

```
class Neo4jService:

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

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

    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)
```

  - **Funciones para crear las relaciones para los nodos**

```
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(seelf, 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_publi_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)
```

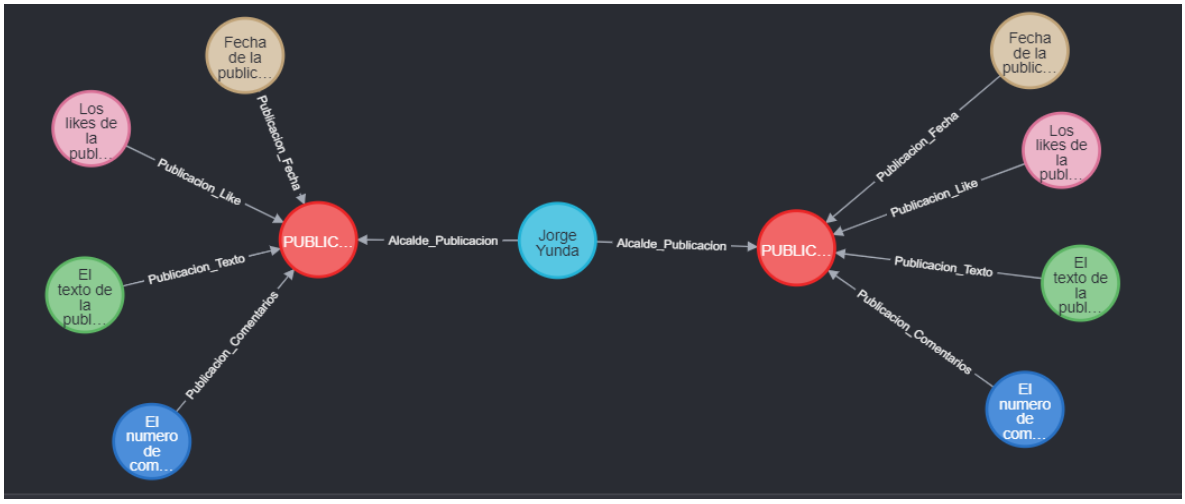
- **For para recorrer las paginas de mi web-scraper de Facebook**

```
neo4j = Neo4jService('bolt://localhost:7687', 'neo4j', 'cuenca')
with neo4j._driver.session() as session:
        contador = -1
        session.write_transaction(neo4j.crear_Alcaldes, "Alcaldes")
        session.write_transaction(neo4j.crear_alcalde, "Jorge Yunda")
        session.write_transaction(neo4j.crear_relacion_Alcs_Alc , "Alcaldes" , "Jorge Yunda")
        for post in get_posts ('jorgeyundamachado', pages=52, timeout=1):
            contador = contador + 1
            publicacion = "PUBLICACION DE YUNDA N°" +str(contador)
            fecha = "Fecha de la publicacion N°" +str(contador)+ " es " +str(post['time'])
            text = "El texto de la publicacion N°" +str(contador)+ " es " +str(post['text'])
            likes = "Los likes de la publicacion N°" +str(contador)+ " es " +str(post['likes'])
            comentarios = "El numero de comentarios de la publicacion N°" +str(contador)+ " es " +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, "Jorge Yunda", publicacion)
            session.write_transaction(neo4j.crear_relacion_publi_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("Termino Yunda")
```

- **Pruebas de terminal que está recorriendo las paginas**

```
PUBLICACION DE YUNDA N°0
PUBLICACION DE YUNDA N°1
PUBLICACION DE YUNDA N°2
PUBLICACION DE YUNDA N°3
PUBLICACION DE YUNDA N°4
PUBLICACION DE YUNDA N°5
PUBLICACION DE YUNDA N°6
PUBLICACION DE YUNDA N°7
PUBLICACION DE YUNDA N°8
PUBLICACION DE YUNDA N°9
PUBLICACION DE YUNDA N°10
PUBLICACION DE YUNDA N°11
PUBLICACION DE YUNDA N°12
PUBLICACION DE YUNDA N°13
PUBLICACION DE YUNDA N°14
PUBLICACION DE YUNDA N°15
```

- **Base de datos**
  - **Grafica de 11 nodos relacionados**



- **Todos los nodos y relaciones que existen**

- **Resumen que da Neo4j**

IAExamen

neo4j (default)

Nodes	1012
Relationships	1011
Labels	7
Relationship Types	6
Property Keys	15

- **Conclusiones**

Como conclusión se llega a que la base de datos no relación Neo4j es una muy buena opción para datos que necesitamos que se representen de modo grafico. La librería de Facebook-Scraper también resulto muy útil en este proyecto debido a que gracias a sus etiquetas nos proporcionan ya la información que deseamos.

- **Recomendaciones**

Como recomendación sugiero antes leer la documentación de la librería Facebook-Scraper para poder realizar con mas eficiencia el Web- Scraping