Taller: Red Social con Neo4j Desktop

Curso: Bases de Datos II
Profesor: Diego Mora

0) Resumen

Objetivo: Modelar una red social en Neo4j Desktop, poblarla con datos y desarrollar una aplicación CRUD que permita manejar usuarios, publicaciones y relaciones de amistad/seguimiento.

Lenguaje/Framework: Libre escogencia (ejemplo: Python Flask/Django, Node.js + Express, Java Spring Boot, etc.).
Entrega: Script complete para recrear base de datos, insertar datos y ejecutar los queries. Aplicación para hacer un CRUD de posteos, usuarios, amistades, etc.

1) Modelo de datos

Nodos obligatorios

- Usuario: id, nombre, email, fechaRegistro

- Publicación: id, contenido, fecha, likes

- Etiqueta: id, nombre

Relaciones obligatorias

(:Usuario)-[:AMIGO_DE]->(:Usuario) (amistad bidireccional)

(:Usuario)-[:SIGUE]->(:Usuario) (seguimiento unidireccional)

(:Usuario)-[:CREA]->(:Publicación)

(:Publicación)-[:TIENE_ETIQUETA]->(:Etiqueta)

Reglas

- Cada Usuario debe tener al menos 1 Publicación.

- Cada Publicación debe tener al menos 1 Etiqueta.

- Usar constraints para claves únicas (Usuario.email, Etiqueta.nombre).

2) Requisitos funcionales

App CRUD

Usuarios:
- Crear usuario (Create)
- Listar usuarios y buscar por nombre (Read)
- Actualizar datos del usuario (Update)
- Eliminar usuario (Delete, con DETACH DELETE)

Publicaciones:
- Crear publicación asociada a un usuario
- Listar publicaciones con filtros (ej. por etiqueta, por usuario)
- Actualizar contenido de publicación
- Eliminar publicación

Relaciones:
- Crear/eliminar amistad
- Crear/eliminar seguimiento

Consultas analíticas

1. Amigos en común entre dos usuarios:
MATCH (u1:Usuario {email:$u1})-[:AMIGO_DE]-(amigo)-[:AMIGO_DE]-(u2:Usuario {email:$u2}) RETURN amigo.nombre

2. Publicaciones más populares (top N por likes):
MATCH (p:Publicación) RETURN p.contenido, p.likes ORDER BY p.likes DESC LIMIT $n

3. Recomendación de amigos:
MATCH (u:Usuario {email:$email})-[:AMIGO_DE]-(amigo)-[:AMIGO_DE]-(sugerencia) WHERE NOT (u)-[:AMIGO_DE]-(sugerencia) AND u<>sugerencia RETURN sugerencia.nombre

3) Población de datos

Mínimos requeridos:
- 15 usuarios
- 30 publicaciones
- 5 etiquetas
- Cada usuario con 2–3 amigos y 3 publicaciones
- Usar MERGE para evitar duplicados

Ejemplo de inserción Cypher:

MERGE (u:Usuario {email:'ana@mail.com'}) SET u.id='U001', u.nombre='Ana', u.fechaRegistro=date('2025-01-01')

MERGE (p:Publicación {id:'P001'}) SET p.contenido='Hola Neo4j!', p.fecha=date('2025-01-10'), p.likes=5

MERGE (u)-[:CREA]->(p)

4) Documentación requerida

1. Script Cypher de creación de nodos, relaciones y constraints.
2. App.

5) Evaluación

Trabajo para realizar en grupos de trabajo final-

Entrega 14 de octubre hasta las 15:00