UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

INGENIERIA EN CIENCIAS DE LA COMPUTACION

Bases de Datos Avanzadas

Sistema de Gestión de Blog

Docente:

José Saúl De Lira Miramontes

Alumnas:

Jazmín Cruz González 367770

Sarahy Chaparro Ramírez 367977

Fecha: 28/11/2025

•

Introducción

Este proyecto consiste en el desarrollo de un sistema web para gestionar un blog, implementando operaciones CRUD (Crear, Leer, Actualizar y Eliminar), utilizando Neo4j, una base de datos orientada a grafos, para modelar las relaciones complejas entre usuarios, artículos, comentarios y categorías de manera eficiente.

Objetivos de aprendizaje:

- Comprender las bases de datos orientada a grafos.
- Implementar el lenguaje de consultas Cypher dentro de una aplicación Python.
- Utilizar Flask como framework web para la interfaz de usuario.
- Modelar nodos y relaciones (:WROTE, :HAS_TAG, :ON_ARTICLE) en lugar de tablas.

Tecnologías Utilizadas

Categoría	Tecnología	Propósito
Backend	Flask(Python)	Framework web
Base de Datos	Neo4j	Base de datos de Grafos
Lenguaje de Consulta	Cypher	Interactuar con los grafos (nodos y aristas).
Driver	neo4j (Python Driver)	Conexión entre Python y Neo4j
Frontend	HTML5 / Jinja2	Interfaz de usuario y plantillas dinámicas.

Justificación:

Neo4j permite manejar datos conectados de forma natural. En un blog, saber "quién escribió qué", "qué comentarios pertenecen a qué artículo" o "qué artículos comparten la misma etiqueta" es mucho más intuitivo y rápido mediante recorridos de grafos.

Arquitectura del sistema

- Modelo: Clase `BlogCompleto` maneja la conexión y operaciones CRUD.
- Vista:Plantillas HTML (interfaz del usuario).
- Controlador: Rutas Flask que gestionan la lógica y peticiones.

Modelo de Grafo

Nodos Principales:

1. User: Representa a los autores y lectores. (Propiedades: uid, name, email).
2. Article: Representa las publicaciones. (Propiedades: uid, title, text).
3. Tag: Etiquetas para clasificar contenido.
4. Category: Categorías generales.
5. Comment: Comentarios de los usuarios.

Relaciones:

- (:User)-[:WROTE]->(:Article): Un usuario escribe un artículo.
- (:Article)-[:HAS_TAG]->(:Tag): Un artículo tiene etiquetas.
- (:Article)-[:IN_CATEGORY]->(:Category): Un artículo pertenece a una categoría.
- (:Comment)-[:ON_ARTICLE]->(:Article): Un comentario está en un artículo.
- (:User)-[:WROTE]->(:Comment): Un usuario escribe un comentario.

Restricciones:

Para asegurar la integridad de los datos, se definieron las siguientes restricciones en Cypher:

```
CREATE CONSTRAINT user_email IF NOT EXISTS FOR (u:User) REQUIRE u.email IS UNIQUE;
CREATE CONSTRAINT user_uid IF NOT EXISTS FOR (u:User) REQUIRE u.uid IS UNIQUE;
CREATE CONSTRAINT article_uid IF NOT EXISTS FOR (a:Article) REQUIRE a.uid IS UNIQUE;
CREATE CONSTRAINT tag_uid IF NOT EXISTS FOR (t:Tag) REQUIRE t.uid IS UNIQUE;
CREATE CONSTRAINT cat_uid IF NOT EXISTS FOR (c:Category) REQUIRE c.uid IS UNIQUE;
```

Implementación del Código

La lógica de la base de datos se encapsuló en la clase BlogCompleto dentro del archivo CRUD.py.

-

Conexión a la Base de Datos

Se utiliza el driver oficial de Neo4j conectando al puerto bolt://localhost:7687.

CRUD

- **Crear usuario:** Crea un nodo User con un UUID único y la fecha de creación

```python
def crear_usuario(self, nombre, email):
    uid = str(uuid.uuid4())
    query = """
    CREATE (u:User {uid: $uid, name: $name, email: $email, created_at: datetime()})
    RETURN u.uid as id
    """
    with self.driver.session() as session:
        result = session.run(query, uid=uid, name=nombre, email=email)
        return result.single()["id"]
```

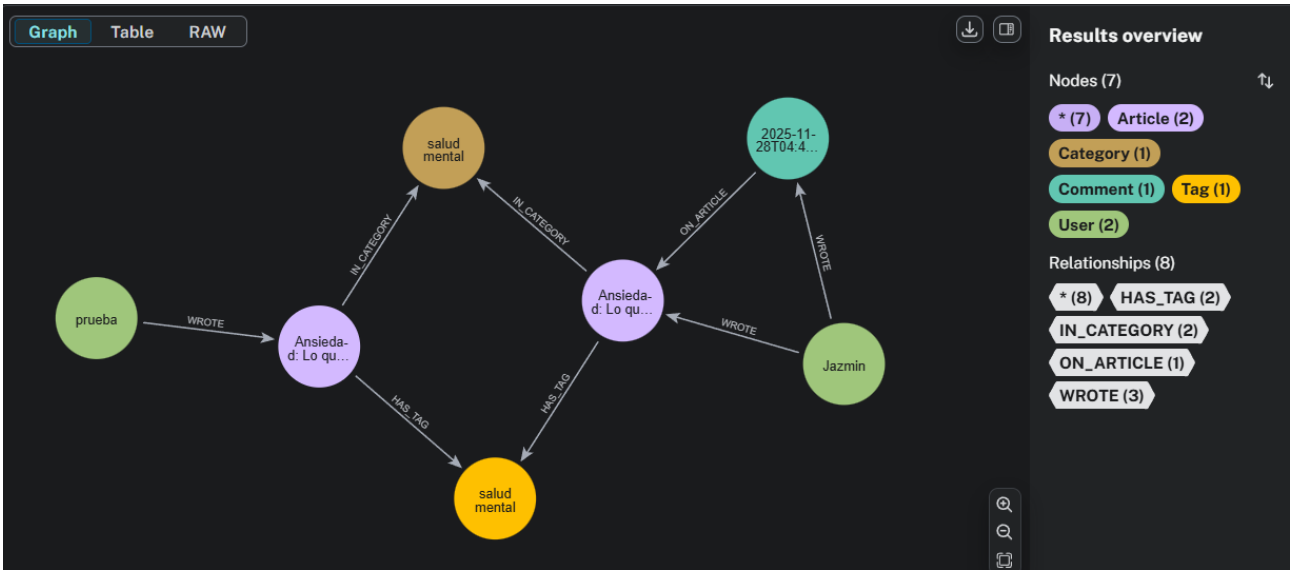
- **Crear articulo:** Se crea un artículo y se conecta al autor mediante la relación WROTE.

```
MATCH (u:User {uid: $autor_id})
CREATE (a:Article {uid: $uid, title: $title, text: $text, date: datetime()})
MERGE (u)-[:WROTE]->(a)
```

- **Crear comentario**

```python
def crear_comentario(self, articulo_id, autor_id, texto):
    uid = str(uuid.uuid4())
    query = """
    MATCH (a:Article {uid: $aid}), (u:User {uid: $uid_user})
    CREATE (c:Comment {uid: $cid, text: $text, date: datetime()})
    MERGE (u)-[:WROTE]->(c)
    MERGE (c)-[:ON_ARTICLE]->(a)
```

Neo4j Browser (Gráfo)



Pruebas y resultados

Comentarios

Añadir Comentario

Artículo:

Ansiedad: Lo que usted debe saber

Autor:

Jazmin

Comentario:

nodanonwcnqocn

Añadir Comentario

Lista de Comentarios

ID	Comentario	Autor	Artículo	Editar/Eliminar

-

Lista de Comentarios

ID	Comentario	Autor	Artículo	Editar/Eliminar
a5b02569-ae37-4c60-ba5a-46272fd41da0	nodanonwcnqocn	Jazmin	Ansiedad: Lo que usted debe saber	Editar | Eliminar

Usuarios

Crear Usuario

Nombre de Usuario:

prueba

Email:

prueba@gmail.com

Crear Usuario

Lista de Usuarios

ID	Nombre	Email	Editar/Eliminar
586da375-a414-49c4-b42c-a27670c10559	Jazmin	jazmincruz15032019@gamil.com	Editar | Eliminar

Lista de Usuarios

ID	Nombre	Email	Editar/Eliminar
586da375-a414-49c4-b42c-a27670c10559	Jazmin	jazmincruz15032019@gamil.com	Editar | Eliminar
1048696b-8787-41d7-8dae-536d7bcf2f3c	prueba	prueba@gmail.com	Editar | Eliminar

Crear Tag

Nombre del Tag:

terror

URL del Tag:

ckjekwlmclka

Crear Tag

Crear Categoría

Nombre Categoría:

películas

URL Categoría:

kjcsnoncoqincnqo

Crear Categoría

Lista de Tags

ID	Nombre	URL	Editar/Eliminar
eab27ef9-cb24-491e-9572-237226b8c3de	salud mental	sznxnxjxdjdxsisncdn	Editar | Eliminar

Lista de Categorías

ID	Nombre	URL	Editar/Eliminar
3bb747a6-2ffd-42cb-9bb8-a68195b52961	salud mental	saludmental-url	Editar | Eliminar

Lista de Tags

ID	Nombre	URL	Editar/Eliminar
eab27ef9-cb24-491e-9572-237226b8c3de	salud mental	sznxnxjxdjdxsisncdn	Editar | Eliminar
a48cfaf5-d575-4799-9a3f-43ad3e816351	terror	ckjekwlmclka	Editar | Eliminar

Lista de Categorías

ID	Nombre	URL	Editar/Eliminar
3bb747a6-2ffd-42cb-9bb8-a68195b52961	salud mental	saludmental-url	Editar | Eliminar
6f165192-54ee-4d36-8fe2-994801e896ca	películas	kjcsnoncoqincnqo	Editar | Eliminar

Artículos

Crear Artículo

Título:

Ansiedad: Lo que usted debe saber

Contenido:

kjjnisaxnixnnas jnsjxniuNXOIUWN

Autor:

prueba

Tags:

- [x] salud mental
- [ ] terror

Categorías:

- [x] salud mental
- [ ] películas

Crear Artículo

-

Lista de Artículos

ID	Título	Autor	Editar/Eliminar
cbf7cea7-3f9c-4f52-bf8b-ebbd7f3c6c56	Ansiedad: Lo que usted debe saber	Jazmin	Editar | Eliminar
a711c95b-d8e7-4c43-a68f-ed5bb3f87d0c	Ansiedad: Lo que usted debe saber	prueba	Editar | Eliminar

Conclusiones

El uso de Neo4j permitió modelar de manera natural las relaciones entre los elementos del blog: usuarios, artículos, comentarios, etiquetas y categorías. A diferencia de MongoDB, los grafos permiten navegar estas relaciones de forma más rápida y eficiente mediante Cypher. Se logró implementar un sistema web completo con Flask y Neo4j, que incluye CRUD para todos los elementos del blog y una interfaz funciona

Aprendizajes

- Uso del driver oficial de Neo4j en Python
- Creación de constraints y modelado de grafos
- Implementación de relaciones como WROTE, HAS_TAG, IN_CATEGORY
- Manejo de sesiones en Neo4j
- Integración Flask ↔ Neo4j
- Diseño de CRUD completos en grafos

Repositorio

https://github.com/sarahy367977/BDA_Proyecto-Final