Ficha Técnica — Módulo 5: Integración Generativa (Ollama + LangChain + Router de Intenciones)

Propósito del componente

Este módulo implementa la integración entre el modelo generativo utilizado (Ollama) y la base de conocimiento representada en Neo4j, mediante un flujo de razonamiento controlado con LangChain y un Router de Intenciones. Su objetivo es permitir que los usuarios interactúen con EduDB utilizando lenguaje natural, mientras el sistema interpreta la pregunta, extrae los parámetros relevantes (esquema, forma normal, tipo de consulta), ejecuta la consulta correspondiente en Neo4j y genera una respuesta explicada.

Este módulo cumple una función clave: convierte texto natural en una consulta estructurada y luego transforma los resultados estructurados en una explicación clara y coherente. Constituye el "cerebro lingüístico" del sistema.

Entradas

El módulo recibe como entradas:

- **Preguntas en lenguaje natural formuladas por el usuario**, tales como:
  - "¿El esquema Pedido cumple 2FN?"
  - "¿Qué se necesita para cumplir 3FN?"
  - "¿Por qué no cumple 2FN?"
  - "Mostrame qué pide la 1FN".

- **Parámetros detectados por el Router** de Intenciones:
  - nombre del esquema
  - forma normal involucrada
  - tipo de intención (estado_fn o requisitos_fn)

- **Conexión al modelo LLM configurado en Ollama**.
- **Conexión a la base de datos Neo4j** para ejecutar la consulta correspondiente.

Salidas

Las salidas de este módulo son:

- **La intención detectada** (estado_fn, requisitos_fn).
- **Los parámetros estructurados extraídos de la pregunta del usuario.**
- **El resultado obtenido desde Neo4j:**
  - cumplimiento o incumplimiento de una FN
  - criterios asociados
  - motivos de violaciones
  - información teórica de una FN específica
- **Una respuesta final explicada en lenguaje natural generada por el LLM.**
- En solicitudes teóricas, **una explicación correcta de los criterios de la FN** (por ejemplo, "Para cumplir 3FN no deben existir dependencias transitivas").

Herramientas utilizadas y entorno

- **Ollama** como entorno de ejecución local del modelo LLM.
- Modelo generativo utilizado: **gpt-oss:120b-cloud.**
- **LangChain** (versión con soporte para langchain_ollama) para definir:

    - el flujo LCEL
    - el Router de Intenciones
    - los Pydantic models utilizados para validar la estructura de las respuestas
- **Archivo .env** para configuración del modelo y credenciales hacia Neo4j AuraDB.
- Integración con el módulo Neo4j mediante consultas construidas dinámicamente.

Arquitectura o funcionamiento interno

El funcionamiento interno del módulo integra tres flujos complementarios:

1. consultas libres en lenguaje natural,
2. consultas teóricas,
3. asistencia dentro del flujo guiado de creación de esquemas.

A. Flujo de consultas libres (preguntas abiertas del usuario)

1. El usuario escribe una pregunta en lenguaje natural.
2. El Router de Intenciones clasifica la consulta en una de dos categorías:
    - estado_fn (preguntas sobre si un esquema cumple o no una FN)
    - requisitos_fn (preguntas conceptuales sobre las condiciones de una FN)
3. El modelo devuelve parámetros estructurados: esquema, forma normal, intención.
4. El sistema genera la consulta Cypher correspondiente.
5. Neo4j retorna datos reales del grafo.
6. El LLM genera una explicación clara para el usuario.

Este flujo se usa para consultas como:
"¿El esquema Pedido cumple 2FN?" o "¿Qué necesita una tabla para cumplir 3FN?"

B. Flujo teórico (consulta de requisitos de una FN)

1. El Router detecta intención requisitos_fn.
2. El LLM produce una estructura con: intención = requisitos_fn, forma_normal = XFN.
3. Neo4j devuelve los criterios asociados a esa forma normal desde el metamodelo.
4. El LLM traduce esos criterios en una explicación comprensible.

Este flujo responde preguntas como:
"Explicame 1FN", "¿Cómo se cumple 2FN?", "¿Qué violaciones rompen 3FN?".

C. Flujo guiado de evaluación de esquemas (interacción paso a paso)

Aunque el motor principal del flujo guiado pertenece al Módulo 7, este módulo participa en dos funciones esenciales:

1. **Interpretación del input del usuario**
   Durante el proceso guiado, el usuario escribe respuestas como:
    - "Sí, tiene clave compuesta"
    - "La clave es IDFactura"
    - "Hay una dependencia transitiva"
    - "Los atributos son fecha, cliente, total"

El LLM clasifica y normaliza esas respuestas para convertirlas en parámetros estructurados.

2. **Estandarización del lenguaje del usuario**
   El modelo limpia y traduce las respuestas del usuario al formato que necesita Neo4j:
    - nombres de atributos

- tipo de dependencia (plena, parcial, transitiva)
- claves primarias
- cantidad de violaciones detectadas

3. **Comunicación con Neo4j**
   Una vez que el usuario definió su esquema paso a paso:
   - el módulo arma la estructura final del esquema,
   - genera los nodos correspondientes (Esquema, Atributos, DF),
   - y activa la evaluación automática posterior de FN.
4. **Devolución de explicaciones**
   El LLM describe el resultado final del flujo guiado en forma clara:
   por ejemplo:
   "El esquema Factura no cumple 3FN porque presenta dependencias transitivas entre IDFactura → IDCliente → NombreCliente."

Este flujo es híbrido: combina interacción natural, extracción estructurada y razonamiento guiado.

Código relevante

El código relevante se compone de:

- definición del modelo LLM utilizando langchain_ollama
- definición del router de intención y los Pydantic models
- prompts específicos para la clasificación
- función route_query() que realiza la inferencia de intención
- integración con el módulo Neo4j

El proyecto está disponible en:
https://github.com/PauRodriguezz/EduDB-ProyectoIA

Ejemplo de instancia dentro del modelo conceptual

Ejemplo 1 — Consulta libre
Usuario: "¿El esquema Pedido cumple 2FN?"

- El Router detecta intención de tipo estado_fn.
- Extrae: esquema = Pedido, forma_normal = 2FN.
- Consulta en Neo4j el estado real: el esquema tiene dependencias parciales, por lo tanto NO cumple.
- El LLM genera una explicación clara y coherente.

Ejemplo 2 — Consulta conceptual
Usuario: "¿Qué se necesita para cumplir 3FN?"

- Intención detectada: requisitos_fn.
- El sistema consulta Neo4j para obtener criterios asociados a 3FN.
- El LLM explica: primero cumplir 1FN y 2FN, no tener dependencias transitivas, etc.

Ejemplo 3 — Consulta guiada
Usuario: carga un nuevo esquema Factura con transitivas presentes.

- El módulo traduce respuestas guiadas a parámetros estructurados.
- Genera nodos en Neo4j.
- El LLM explica por qué no cumple 3FN.

Resultados obtenidos

- Las respuestas generadas por el modelo fueron claras, correctas y consistentes con el grafo.
- Se validó que el modelo no "inventa" datos al realizar consultas basadas en Neo4j.
- La integración demostró robustez tanto para consultas teóricas como aplicadas.
- El flujo guiado y el flujo libre funcionaron sin inconsistencias.

Este módulo habilita una interacción completamente natural con el sistema y permite que la complejidad de las reglas internas quede ocultada al usuario.

Observaciones y sugerencias

- Sería conveniente ampliar el router con nuevas intenciones: claves, dependencias, sugerencia de refactorización de esquemas.
- Se podría agregar un modo de explicaciones paso a paso ("modo detallado").
- En el futuro, podría incorporarse memoria conversacional o recuperación de contexto.
- Sería beneficioso permitir retroalimentación del usuario ("¿te fue útil esta respuesta?") para ajustar los prompts.