

Conversación con Gemini

Exported: 15/6/2026 9:04:17

Jordi H.

Has dicho necesito explorar la base de datos "Linked Data" de la Union Europea. puedes ayudarme?

Gemini

Explorar la base de datos de **Linked Data** (Datos Enlazados) de la Unión Europea es una excelente iniciativa, especialmente si trabajas en analítica y arquitectura de sistemas. La principal puerta de entrada a este ecosistema es el portal **data.europa.eu**, que unifica el antiguo portal de datos abiertos de la UE con el portal de datos europeo. Aquí tienes los recursos clave y la metodología para empezar a trabajar con ellos, priorizando el uso de software libre y herramientas programáticas como Pyt

1. El Endpoint SPARQL (El corazón del LOD)

Toda la metadata del portal se almacena como triplets RDF y es consultable mediante **SPARQL**. Este es el punto de partida para cualquier arquitecto de datos

- Interfaz Web: data.europa.eu/sparql
- Endpoint para API/Python: <https://data.europa.eu/sparql>

Ejemplo de consulta rápida

Para familiarizarte, puedes ejecutar esta consulta básica que lista los primeros 10 datasets registrados en el portal: Fragmento de código

CODE

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?title
WHERE {
  ?dataset a dcat:Dataset .
  ?dataset dct:title ?title .
  FILTER (lang(?title) = 'es') # Filtrar por español
}
LIMIT 10
```

2. Otros Repositorios Especializados

Dependiendo de qué tipo de datos busques, existen otros endpoints con información más granular:

- Cellar (Publicaciones Oficiales): Es el repositorio de la Oficina de Publicaciones de la UE. Contiene tratados, legislación (EUR-Lex) y bibliografía. Endpoint: <https://publications.europa.eu/webapi/rdf/sparql>
- Eurostat (Clasificaciones): Ideal para trabajar con nomenclaturas como NUTS (regiones) o NACE (actividades económicas). Herramienta: ShowVoc (para navegar vocabularios visualmente)

3. Acceso desde Python (Tu Stack Principal)

Como utilizas Python, te recomiendo usar la librería **RDFLib** o simplemente requests para consumir los

endpoints. Aquí tienes un esquema de cómo podrías integrarlo en tu laboratorio o servidor:Pytho

CODE

```
import requests

def query_eu_data(sparql_query):
    url = "https://data.europa.eu/sparql"
    params = {
        'query': sparql_query,
        'format': 'application/sparql-results+json'
    }
    response = requests.get(url, params=params)
    return response.json()

# Tu consulta SPARQL aquí
query = """
SELECT * WHERE { ?d a <http://www.w3.org/ns/dcat#Dataset> } LIMIT 5
"""

data = query_eu_data(query)
print(data)
```

4. Recursos para tu Homelab y Aprendizaje

Dado que estás construyendo un servidor como vector tecnológico, te sugiero estos pasos para profundizar:

- Modelado DCAT-AP: Los datos de la UE siguen el estándar DCAT Application Profile. Entender este esquema es vital para la arquitectura de cualquier sistema que consuma estos datos
- Triple Stores de Software Libre: Si quieres replicar parte de estos datos en tu homelab, puedes experimentar montando un contenedor con Apache Jena Fuseki o GraphDB (versión Free), que te permitirán gestionar tus propias bases de datos de grafo
- data.europa academy: Tienen cursos gratuitos sobre Linked Data y SPARQL específicamente diseñados para sus portales

¿Hay algún tema específico de la Unión Europea (economía, medio ambiente, legislación) sobre el cual te gustaría que construyéramos una consulta más compleja

Jordi H.

Has dicho sobre la investigación científica

Gemini

Para explorar la investigación científica dentro del ecosistema de **Linked Data** de la Unión Europea, el recurso más potente es el **OpenAIRE Graph**, que está integrado en el portal de datos de la UE pero tiene su propio modelo de datos enlazados. Aquí te detallo cómo abordar este dominio desde una perspectiva de arquitectura de datos y software lib

1. El Grafo de OpenAIRE (Research Graph)

Este es uno de los mayores conjuntos de datos enlazados sobre ciencia en el mundo. Conecta publicaciones, conjuntos de datos, software de investigación, proyectos (como Horizon Europe) y organizaciones

- Modelo de Datos: Utiliza una ontología basada en el estándar CERIF (Common European Research Information Format)
- Acceso SPARQL: Aunque OpenAIRE ofrece APIs REST robustas, su integración con la UE permite trazar el impacto de la financiación pública en la producción científica

- **Software Libre Recomendado:** Para procesar estos grafos masivos, puedes usar NetworkX en Python para análisis de topología de redes científicas

2. CORDIS: Proyectos de I+D de la UE

Si tu interés es la estructura de los proyectos y quién recibe la financiación, la base de datos **CORDIS** es la referencia. Sus datos están disponibles como LOD (Linked Open Data)

- **Vocabularios:** Utiliza la ontología EuroVoc para clasificar los campos científicos (FOS - Fields of Science).
- **Arquitectura de consulta:** Puedes cruzar datos de CORDIS con el registro de organizaciones (ORG ID) para ver qué universidades o empresas de tecnología están liderando la investigación en arquitecturas de servidores o Python

Consulta SPARQL de ejemplo (CORDIS)

Esta consulta busca proyectos financiados bajo el programa "Horizon" relacionados con "Artificial Intelligence": Fragmento de código

CODE

```
PREFIX ec: <http://data.europa.eu/949/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?project ?title ?objective
WHERE {
  ?project a ec:Project ;
  ec:title ?title ;
  ec:objective ?objective .
  FILTER(contains(lower(?objective), "artificial intelligence"))
}
LIMIT 5
```

3. Integración en tu Homelab/Laboratorio

Para un analista de datos con un enfoque en arquitectura, el reto de estos datasets es su **volumen y edad**. Aquí algunas recomendaciones para tu infraestructura:

- **Ingesta:** No intentes descargar todo el grafo de una vez. Utiliza el protocolo OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), muy común en ciencia, para traer metadatos a tu servidor local
- **Almacenamiento:** Dado que buscas una solución de software libre, Virtuoso Open-Source Edition es el estándar de la industria para alojar réplicas de estos Triple Stores de la UE
- **Procesamiento:** Usa PySpark si vas a manejar los volcados completos de OpenAIRE (que suelen venir en JSON-L o RDF), ya que la relación entre investigadores y publicaciones genera grafos de miles de millones de aristas

4. Visualización y Análisis Científico

Para descubrir "vectores tecnológicos" (áreas con crecimiento acelerado), puedes usar:

1. **VOSviewer:** Herramienta gratuita para construir y visualizar redes bibliométricas.
2. **Bibliometrix (R/Python):** Librerías de código abierto para mapear la evolución de campos científicos específicos

Un punto crítico de reflexión

La base de datos de la UE es excelente para la **metadata** (quién, cuándo, con qué dinero), pero para el **contenido** (el texto de los papers), deberás conectar estos enlaces con repositorios como **arXiv** o **DOAJ** (Directory of Open Access Journals). ¿Te interesaría que profundizáramos en cómo extraer los nodos de

"Software de Investigación" (repositorios de GitHub/GitLab vinculados a papers) usando Python?

Jordi H.

Has dicho quisiera ir construyendo una herramienta que pueda responder preguntas sobre gestión de la investigación científica. Mi idea es expandir el sistema de información de mi universidad (personas, proyectos, convocatorias, financiación, equipos, departamentos, publicaciones) con entidades homólogas europeas (otras universidades, empresas, institutos) que estén relacionadas por la financiación, por la colaboración o que puedan ser candidatas a través de intereses comunes. Uno de los primeros hitos que deseo conseguir es tener un registro de toda la información que tiene la UE sobre los proyectos de mi universidad. El problema a que quiero solucionar es los gaps de información que tiene mi sistema comparado con lo que tiene registrado la UE en su Link

Gemini

Este es un proyecto de arquitectura de datos ambicioso y extremadamente valioso. Estás planteando esencialmente un **Grafo de Conocimiento Institucional** federado con el **European Research Area (ERA)**. Para resolver los gaps de información entre tu universidad y la Unión Europea utilizando Linked Data, el enfoque debe ser el **mapeo de identidades**. El problema no suele ser la falta de datos, sino la falta de identificadores únicos que conecten ambos sistemas. Aquí tienes una hoja de ruta técnica para conseguir ese primer hito

1. El Identificador Maestro: PIC (Participant Identification Code)

La Unión Europea identifica a todas las instituciones mediante el **PIC**. Este es el "pegamento" que unirá tu sistema local con el Linked Data europeo

- Acción: Localiza el PIC de tu universidad (puedes buscarlo en el Funding & Tenders Portal).
- En Linked Data: En el endpoint de la UE (CORDIS), las organizaciones suelen estar mapeadas con este código o mediante URIs de la oficina de publicaciones

2. Extracción de Proyectos (Hito 1) en Python

Como eres analista de datos y usas Python, te recomiendo crear un script que actúe como "Sincronizador". Vamos a atacar el endpoint de **CORDIS** para extraer todo lo que la UE dice que tu universidad ha hecho

Consulta SPARQL Propuesta

Esta consulta busca proyectos donde tu universidad (identificada por su nombre o PIC) es participante o coordinadora: Fragmento de código

CODE

```
PREFIX ec: <http://data.europa.eu/949/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?project ?projectTitle ?role ?startDate ?endDate ?totalCost
WHERE {
  ?project a ec:Project ;
  ec:title ?projectTitle ;
  ec:hasParticipation ?participation .

  ?participation ec:hasOrganisation ?org ;
  ec:role ?role . # Coordinador o Participante

  ?org rdfs:label ?orgName .

  # Filtro por el nombre de tu universidad (o usa su URI si la conoces)
```

```
FILTER(CONTAINS(LCASE(?orgName), "nombre de tu universidad"))
```

```
OPTIONAL { ?project ec:startDate ?startDate }  
OPTIONAL { ?project ec:endDate ?endDate }  
OPTIONAL { ?project ec:totalCost ?totalCost }  
}
```

3. Arquitectura para la "Detección de Gaps"

Para tu laboratorio/servidor, te sugiero un flujo de **Data Quality** basado en software libre:

1. Capa de Ingesta (Python): Un script que extraiga los datos de tu universidad (vía SQL o API) y los datos de la UE (vía SPARQL)
2. Capa de Alineación (OpenRefine): OpenRefine es una herramienta de software libre increíble para la reconciliación de datos. Tiene un servicio de reconciliación para Wikidata y puedes configurarlo para SPARQL. Te permitirá ver dónde los nombres de los proyectos no coinciden exactamente
3. Capa de Almacenamiento (GraphDB/Fuseki): No almacenes esto en SQL. Usa una base de datos de grafos para mantener la naturaleza "Linked" de los datos. Podrás tener nodos LocalProject y EUPProject y crear una arista sameAs cuando confirmes que son el mismo

4. Entidades Homólogas y Colaboraciones

Una vez tengas los proyectos limpios, el sistema se expande solo gracias al grafo:

- Colaboradores: Si el Proyecto A tiene a tu universidad y a la Universidad de Berlín, ya tienes el nodo "Berlín" y la relación "Colaborador en Proyecto A"
- Intereses Comunes: A través de los códigos EuroVoc o los H2020/Horizon Topics asociados a los proyectos, puedes buscar en el Linked Data qué otras empresas están ganando proyectos en esos mismos topics pero en los que tu universidad aún no participa

5. Próximo paso técnico

Para ayudarte con el código exacto, ¿podrías confirmarme si tienes acceso a los datos de tu universidad en algún formato estructurado (CSV, SQL, API)? Si me das el nombre de tu universidad (o el de una similar para pruebas), puedo generarte un script de Python completo que haga la consulta, limpie los resultados y te genere un reporte de "Proyectos encontrados en la UE vs. Proyectos locales"

Jordi H.

Has dicho tengo acceso a los datos en SQL, Mongo MQL, CSV y directamente a la API de mi sistema de información. hablando de IT, haz una propuesta de qué software debo instalar para tener una plataforma sólida y versátil. Deberé familiarizarme con ella tanto a nivel de mantenimiento del sistema como de uso y organización de los datos (triplet)

Gemini

Para construir una plataforma de **Linked Data** sólida, versátil y basada en software libre que actúe como tu "vector tecnológico", necesitas una arquitectura que separe la **ingesta/transformación** de **almacenamiento/consulta**. Dado que tienes un perfil de arquitecto de sistemas y analista, esta es la propuesta de stack profesional para tu servidor/home:

1. El Motor de Base de Datos (Triple Store)

Necesitas un lugar donde las tripletas RDF "vivan" y se puedan consultar con SPARQL.

- Opción Recomendada: GraphDB Free (o Apache Jena Fuseki si prefieres 100% Open Source puro). Por qué: Aunque la versión Free no es código abierto, es el estándar de facto en la industria. Permite inferencia (razonamiento lógico), visualización de grafos integrada y una API REST excelente para tu Python. Mantenimiento: Se despliega fácilmente en un contenedor Docker. Gestión de memoria Java (JVM) es el punto crítico a
- Alternativa 100% FOSS: Virtuoso Open-Source Edition. Por qué: Es el que usa la Unión Europea y DBpedia. Es increíblemente rápido y soporta SQL y RDF simultáneamente. Es más complejo de configurar, pero muy potente

2. Capa de Ingesta y ETL Semántico

Aquí es donde conectarás tus fuentes (SQL, Mongo, API) para convertirlas en tripletas.

- RMLMapper / Morph-KGC: Función: Morph-KGC es una librería de Python (ideal para ti) que implementa el lenguaje RML. Permite definir reglas para mapear una tabla SQL o un JSON de una API a tripletas RDF sin escribir código de transformación complejo
- Airflow: Función: Para orquestar tus tareas. Un DAG de Airflow puede: 1. Consultar tu API, 2. Lanzar Morph-KGC, 3. Cargar las tripletas resultantes en GraphDB

3. Herramienta de Reconciliación (Crucial para tu Hito 1)

- OpenRefine: Función: Es la navaja suiza para limpiar datos. Tiene extensiones para RDF y permite conectar con servicios de reconciliación. Lo usarás para decir: "Este nombre de proyecto en mi SQL es el mismo que esta URI en el Linked Data de la UE"

4. Capa de Aplicación / Interfaz

- Streamlit: Función: Para crear el frontend de tu herramienta de gestión de investigación de forma rápida usando solo Python. Puedes crear dashboards que consulten a tu GraphDB vía SPARQL y muestren los "gaps" de información
- Yasgui: Función: Una interfaz web para que tú mismo lances consultas SPARQL con autocompletado y visualización de tablas

Propuesta de Arquitectura (Docker Setup)

Te recomiendo organizar tu servidor con docker-compose. Este sería tu ecosistema:

1. Contenedor graphdb: Tu cerebro de datos (Puerto 7200).
2. Contenedor postgres: (Opcional) Como caché local para tus datos de la universidad si no quieres atacar la API en tiempo real
3. Contenedor jupyter-lab: Tu entorno de desarrollo donde usarás rdflib y SPARQLWrapper en Python para experimentar
4. Contenedor morph-kgc: Para ejecutar los mapeos de tus datos locales a RDF.

Tu primera tarea de aprendizaje (Organización de Tripletas)

Para que el sistema sea versátil, no puedes simplemente volcar datos. Debes familiarizarte con estas

Ontologías de Referencia para investigación

1. VIVO Ontology: El estándar para representar personas, organizaciones y actividades académicas.
2. DCAT-AP: Es el que usa la UE. Si tus datos locales "hablan" DCAT, la comparación con la UE será automática
3. FOAF (Friend of a Friend): Para las relaciones entre investigadores y equipos.

Siguiente paso recomendado:

Instala **Docker** y despliega un contenedor de **GraphDB Free**. Una vez lo tengas, puedo pasarte un archivo de

mapeo RML básico para que transformes un CSV de tus proyectos locales a un formato que la UE "entienda"
. ¿Te parece bien empezar por ah
