# **Trabajo Práctico Integrador Base de datos**

**Tema:** Sistema de Gestión y Recomendación de Viajes  
**Bases de datos a utilizar:** Neo4j, MongoDB, Redis.  
**Integrantes:** Nogueira Santiago

**1. Introducción**

El presente trabajo práctico tiene como objetivo desarrollar un entorno en **JupyterLab** que integre distintos tipos de bases de datos para modelar un **Sistema de Gestión y Recomendación de Viajes**.  
El sistema debe permitir **almacenar información sobre usuarios, destinos, hoteles, actividades y reservas**, así como **realizar recomendaciones personalizadas** basadas en datos distribuidos entre **Neo4j, MongoDB y Redis**.

En el mundo actual, los viajes y el turismo generan grandes volúmenes de información sobre usuarios, destinos, alojamientos y actividades. Las empresas del sector necesitan sistemas capaces no solo de registrar estos datos, sino también de **analizarlos y explotarlos** para ofrecer experiencias más personalizadas y eficientes.

Este proyecto busca representar de forma práctica cómo integrar diferentes tipos de bases de datos para gestionar y analizar esa información de manera distribuida y coherente.

**2. Objetivos del proyecto**

El sistema debe permitir:

* Almacenar información de usuarios, historial de reservas, destinos turísticos, hoteles, actividades y precios asociados.
* Gestionar datos temporales como búsquedas recientes o reservas en proceso.
* Representar relaciones de conocimiento entre usuarios (amistades, parentescos) y vínculos entre usuarios y destinos.
* Proveer una estructura modular que integre las tres bases de datos y facilite consultas combinadas.

**3. Decisiones de almacenamiento**

La principal decisión del equipo fue **cómo utilizarlas de forma complementaria**, definiendo qué tipo de información debía almacenarse en cada una y cómo coordinar su funcionamiento.

**MongoDB — Datos estructurados y descriptivos**

Se utilizó **MongoDB** para almacenar la información base del sistema, representada en formato documental JSON.  
Esto incluye:

* usuarios.json — Datos de usuarios (nombre, edad, país, preferencias).
* destinos.json — Información de destinos turísticos (ciudad, país, descripción, atractivos).
* hoteles.json — Hoteles asociados a cada destino (nombre, ciudad, precio, calificación).
* actividades.json — Actividades disponibles por destino o ciudad.
* reservas.json — Reservas realizadas por los usuarios.

MongoDB es adecuado para estos datos porque permite almacenar estructuras flexibles y heterogéneas, manejar listas (por ejemplo, servicios de hotel o intereses del usuario) y realizar consultas por filtros o agregaciones (como “destinos más reservados” o “promedio de precios por ciudad”).

**🕸️ Neo4j — Relaciones sociales y de viaje**

**Neo4j** se utilizó para modelar las relaciones entre entidades, ya que su estructura de grafos es ideal para representar conexiones y asociaciones.

Los archivos CSV (relaciones\_sociales.csv y visitas.csv) contienen información para crear los nodos y relaciones del grafo:

* (:Usuario)-[:AMIGO\_DE]->(:Usuario)
* (:Usuario)-[:FAMILIAR\_DE]->(:Usuario)
* (:Usuario)-[:VISITÓ]->(:Destino)

Esto permite consultas complejas como:

* “Amigos de un usuario que visitaron los mismos destinos.”
* “Destinos recomendados según la red de contactos.”

El uso de Neo4j facilita la implementación de algoritmos de recomendación basados en relaciones sociales y coincidencias de comportamiento.

**⚡ Redis — Datos temporales y de sesión**

**Redis** se empleó para gestionar información de carácter **temporal o volátil**, por ejemplo:

* Sesiones de usuario activas.
* Reservas en proceso.
* Resultados de búsqueda cacheados.

Redis, al ser una base de datos en memoria, ofrece un acceso extremadamente rápido y la posibilidad de establecer tiempos de expiración automática.  
De este modo, evita sobrecargar las otras bases de datos con información transitoria.

**4. Estructura general del proyecto**

📦 sistema-viajes/

│

├── 🧠 notebooks/

│ └── Notebook\_Principal.ipynb # Ejecución principal del trabajo

│

├── 📂 data/

│ ├── usuarios.json

│ ├── destinos.json

│ ├── hoteles.json

│ ├── actividades.json

│ ├── reservas.json

│ ├── relaciones\_sociales.csv

│ └── visitas.csv

│

├── 🐳 docker/

│ ├── Dockerfile

│ ├── requirements.txt

│ ├── docker-compose.yml

│ └── .env

│

├── 📜 scripts/

│ ├── config\_paths.py

│ ├── funciones\_comunes.py

│ ├── funciones\_mongo.py

│ ├── funciones\_neo4j.py

│ ├── funciones\_redis.py

│

├── 📊 Otros/

│ ├── consigna TP.pdf

│ └── …

│

├── 📘 README.md

├── .gitignore

└── 📦 entrega\_final.zip

**5. Esquema conceptual**

USUARIO ───< REALIZA >─── RESERVA ───< CORRESPONDE\_A >─── DESTINO

│

├──< AMIGO\_DE / FAMILIAR\_DE >── USUARIO

│

└──< VISITA >── DESTINO ───< TIENE >── HOTEL

│

└──< OFRECE >── ACTIVIDAD

Este modelo refleja tanto la estructura relacional (para Neo4j) como la organización documental (para MongoDB).

**6. Interacción entre archivos y servicios**

┌─────────────────────────┐

│ Notebook\_Principal.ipynb│

│ (orquestador general) │

└──────────┬──────────────┘

│

┌─────────────────┼────────────────────┐

│ │ │

┌──────────▼────────┐ ┌──────▼──────────┐ ┌───────▼─────────┐

│ funciones\_mongo.py│ │ funciones\_neo4j │ │ funciones\_redis │

│ (CRUD + consultas)│ │ (nodos y rel.) │ │(cache, sesiones)│

└──────────┬────────┘ └────────┬────────┘ └────────┬────────┘

│ │ │

┌────────▼────────┐ ┌──────▼──────────┐ ┌──────▼────────┐

│ MongoDB │ │ Neo4j │ │ Redis │

│ (colecciones) │ │ (nodos y rel.) │ │ (memoria) │

└─────────────────┘ └─────────────────┘ └───────────────┘

▲ ▲

│ │

┌──────────┴──────────┐ ┌─────┴────────────┐

│ funciones\_comunes.py│ │ funciones\_comunes│

│ (lectura JSON/CSV) │ │ (lectura CSV) │

│ JSON → MongoDB │ │ CSV → Neo4j │

└─────────────────────┘ └──────────────────┘

El **Notebook principal** actúa como **orquestador** del sistema:  
lee los datos, ejecuta las funciones de carga, gestiona la conexión a cada base de datos y permite realizar consultas integradas.

**7. Conclusiones**

El modelo propuesto cumple con el objetivo de integrar distintos tipos de bases de datos dentro de un mismo entorno analítico.  
Si bien las tecnologías fueron asignadas por la cátedra, el diseño de su **interacción y uso complementario** fue una decisión del grupo, orientada a representar de forma coherente las distintas dimensiones del sistema de viajes.

El resultado es un entorno modular, escalable y con una estructura clara, que permite tanto la carga de datos como la ejecución de consultas complejas y análisis integrados entre distintas fuentes.