



## Tarea 3: MongoDB INF-239 Bases de Datos

*Profesores: Ricardo Salas (ricardo.salas@usm.cl)*

*Mauricio Figueroa (mauricio.figueroa@usm.cl)*

*Ayudante Coordinador: Gonzalo Alarcón (gonzalo.alarcon@usm.cl)*

*Noviembre, 2025*

### 1 Objetivo

Investigar y aplicar los conceptos básicos sobre el gestor de base de datos NoSQL MongoDB, utilizando herramientas como Docker para desplegar los servicios, MongoDB Query Language (MQL) y Python.

### 2 Especificaciones y reglas

El desarrollo de esta tarea debe cumplir las siguientes especificaciones, de lo contrario existiría un descuento en la nota final:

- Se debe ejecutar todos los procedimientos necesarios para implementar la BD MongoDB sobre Docker, bajo una arquitectura distribuida maestro-esclavo, utilizando Python como lenguaje de programación para realizar las operaciones que se requieran en esta tarea.
- La tarea debe ser entregada el 17 de noviembre hasta las 23:59 hrs., cuya respectiva defensa es obligatoria.

#### 2.1 Entregables

La tarea debe ser entregada como un archivo comprimido de la forma T3-ROL1-ROL2.zip y debe contener los siguientes archivos:

1. Archivo deploy-rol1-rol2.yml utilizado para desplegar los servicios sobre Docker con las especificaciones de arquitectura entregadas.
2. Archivo Jupyter Notebook o Google Colab pymongo-rol1-rol2.ipynb, el cuál debe venir ejecutado con todas las evidencias de los resultados documentadas en el mismo notebook, considerando una sección por cada requerimiento solicitado en la tarea.
3. Archivo discursos-final-rol1-rol2.json que contenga los datos actualizados y finales que den cuentas de las operaciones realizadas según los requerimientos de esta tarea.
4. Archivo README con las indicaciones de uso y supuestos utilizados en su tarea.

#### 2.2 Reglas

- El trabajo debe realizarse en parejas; no se aceptarán tareas individuales. Las copias serán evaluadas con nota 0 y se informará a las autoridades correspondientes. Para consultas, utilicen la plataforma oficial AULA, hasta 48 horas antes del plazo de entrega original
- En el foro de búsqueda de pareja podrán buscar compañeros quienes no tengan, siendo esta una responsabilidad exclusiva del estudiante. En caso de problemas con su pareja, podrán contactar al profesor explicando su situación. Solo un alumno debe realizar la entrega.
- Si falla la ejecución de algún comando, no se asignará puntaje a este. Existe la posibilidad de que, durante su defensa, asista su profesor y realice preguntas. Es responsabilidad del estudiante inscribir un horario de defensa y estar presente en la fecha y hora elegida.

- Cada grupo tendrá un horario definido para su defensa; en caso de atraso, contarán con un tiempo menor para presentar su trabajo. La información respecto a la defensa se publicará eventualmente en AULA, lo que incluirá detalles sobre los descuentos. Es su obligación estar atentos a esta información y cumplir con lo establecido allí.

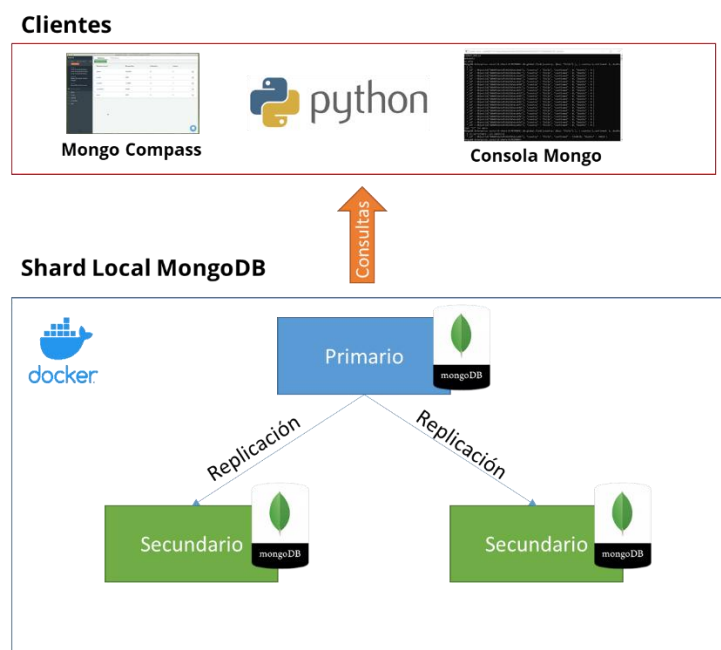
### 3 CONTEXTO

En el marco del avance de los sistemas inteligentes, las búsquedas tradicionales han dado paso a arquitecturas más sofisticadas como los sistemas **RAG (Retrieval-Augmented Generation)**, que integran modelos de lenguaje con bases de datos vectoriales para realizar búsquedas semánticas (Lewis et al., 2020).

Este laboratorio tiene como objetivo simular una base de datos vectorial sobre una infraestructura local de **MongoDB** configurada como Replica Set en Docker. Se trabajará con un corpus de discursos históricos que será procesado mediante herramientas de NLP para su vectorización (*embeddings*) (Mikolov et al., 2013). A través de esta experiencia, los estudiantes conocerán en profundidad los principios de bases de datos documentales, indexación semántica, recuperación de información por similitud y alta disponibilidad.

### 4 REQUISITOS

1. Implementar la arquitectura de *Cluster* con un set de 3 réplicas. Una de las máquinas será el primario y las otras dos serán los secundarios. Esta configuración deberá estar operativa en su máquina local, donde se levantarán las 3 instancias de **Docker** con cada uno de los servicios de **MongoDB** con la redundancia de la Base de Datos llamada “**Política**” que contendrá la colección de documentos llamada “**Discursos**”. En la siguiente imagen se muestra la arquitectura requerida.



**Figura 2:** Arquitectura General



2. Realiza el preprocesamiento del corpus textual considerando lo siguiente:

- Utilizar un corpus<sup>1</sup> de discursos históricos en formato .txt entregado por el docente (**DiscursosOriginales.rar**).
- Para cada documento:
  1. Calcular su hash SHA-256 (Python SHA256, 2023) y usarlo como “\_id”.
  2. Generar su embedding (vector) utilizando librerías como sentence-transformers (sentence transformer documentation, nd) o spaCy (Spacy, nd).
- Insertar el documento en MongoDB con la siguiente estructura:

```
{
  "_id": "<hash SHA-256>",
  "texto": "<contenido completo>",
  "embedding": [ ... ],
}
```

3. Desarrollar un script en Python que:

- Reciba una consulta textual.
- Genere su embedding.
- Calcule la similitud coseno (Cosine\_similarity, n.d.) entre el embedding de consulta y los embeddings almacenados.
- Devuelva los top 5 documentos más similares, ordenados por mayor similitud, mostrando metadatos y puntaje.
- Mostrar resultados por consola o Jupyter Notebook.

---

<sup>1</sup> Según la RAE, **Corpus**: Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación.



#### 4. Demostrar la Robustez del Sistema

Una vez realizados los puntos anteriores, realizando operaciones necesarias desde Python, se pide mostrar evidencia objetiva que permita demostrar:

- a) La consistencia de los datos en los 3 nodos replicados, por ejemplo, posterior a la inserción, actualización y/o eliminación de datos, los datos deben permanecer siempre consistentes.
- b) La alta disponibilidad, por ejemplo, bajando el nodo local para que responda alguno de los nodos replicados sin que el usuario se percate del problema.

#### 5. Consideraciones técnicas

Para llevar a cabo esta tarea, se recomienda llevar a cabo las siguientes actividades técnicas:

1. Instalar el software de desarrollo y tecnología para contenedores Docker Desktop y Docker Compose. Usar como referencia este sitio <https://docs.docker.com/manuals/>.
2. Crear un Replica Set de MongoDB en Docker con docker-compose
3. Conectarse al Replica Set desde Python usando la biblioteca PyMongo, con la dirección de conexión de ejemplo `mongodb://mongo1:30001,mongo2:30002,mongo3:30003/?replicaSet=my-replicaset&readPreference=primary&appname=MongoDB%20Compass&ssl=false`. Asegúrese de modificar el archivo de hosts para que los nombres de los servidores sean mapeados a la IP local del host.
4. Descargar el dataset DiscursosOriginales.rar
5. Documentación acerca de MongoDB, su arquitectura y su lenguaje de consultas puede ver en <https://www.mongodb.com/docs/manual/>.



## 5 REFERENCIAS BIBLIOGRÁFICAS

1. Cosine\_similarity. (n.d.). Scikit-Learn. Retrieved May 16, 2025, from [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine\\_similarity.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html).
2. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459–9474. [https://papers.nips.cc/paper\\_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf](https://papers.nips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf)
3. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781. <https://arxiv.org/abs/1301.3781>
4. Python SHA256: Secure hashing implementation. (2023, December 4). Medium. <https://medium.com/@wepypixel/python-sha256-secure-hashing-implementation-pypixel-7b8434a9b244>
5. SentenceTransformers Documentation — Sentence Transformers documentation. (n.d.). Sbert.net. Retrieved May 16, 2025, from <https://sbnet.net/>
6. SpaCy · industrial-strength Natural Language Processing in Python. (n.d.). Spacy.io. Retrieved May 16, 2025, from <https://spacy.io/>