Licenciatura en Sistemas de Información
Bases de Datos NSQL

PRACTICA #6

API Aeropuertos

El proyecto debe incluir:

- MongoDB → Almacenamiento principal de los datos de aeropuertos.
- Redis GEO → Ubicación de aeropuertos para cálculos de cercanía.
- Redis Popularidad → Ranking de aeropuertos más visitados.
- Backend → API REST.
- Frontend (Leaflet.js) → Visualización en mapa con clustering.
- Docker Compose → Orquestación de todos los servicios.

Carga Inicial de Datos:

- El proyecto debe incluir un archivo `airports.json` con datos de aeropuertos.
- Al iniciar el sistema por primera vez, debe ejecutarse un proceso que:
- Cargue los datos en **MongoDB** (`airports` collection).
- Agregue cada aeropuerto a **Redis GEO** con `GEOADD`.
- Opcionalmente, prepare Redis Popularidad (`ZSET` vacío).

Este proceso debe automatizarse en el arranque del backend (o como tarea manual ejecutable desde el contenedor).

API REST – Funcionalidades Mínimas
CRUD de Aeropuertos

- `POST`
  `/airports`
  → Crea un nuevo aeropuerto: lo guarda en MongoDB y lo agrega a Redis GEO.
- `GET`
  `/airports`
  → Devuelve la lista de todos los aeropuertos.
- `GET`
  `/airports/{iata_code}`
  →Devuelve los datos de un aeropuerto.
  → **Suma +1 en Redis Popularidad.**
- `PUT`
  `/airports/{iata_code}`
  → Modifica los datos del aeropuerto (MongoDB).
- `DELETE`
  `/airports/{iata_code}`
  → Elimina el aeropuerto de MongoDB, Redis GEO y Redis Popularidad.

Licenciatura en Sistemas de Información
Bases de Datos NSQL

Consultas Geoespaciales

- `GET /airports/nearby?lat=..&lng=..&radius=km`

  → Busca aeropuertos cercanos usando `GEORADIUS` en Redis.

Estadísticas de Popularidad

- `GET /airports/popular`

  → Devuelve los aeropuertos más accedidos (usando `ZREVRANGE` de Redis Popularidad).

Redis GEO

- Cada aeropuerto debe guardarse con su coordenada (`GEOADD airports-geo lng lat IATA`).
- Se usará para responder a `/airports/nearby`.

Redis Popularidad

- Cada vez que se accede a `/airports/{iata_code}`, se suma +1 en el `ZSET. ZINCRBY testset 1 member`
- `obtener los mas populares zrange testset 0 10 rev withscores`

`Se debe implementar` **`expiración`** `automática de este set:`

- TTL de 1 dia (`EXPIRE airport_popularity 86400`)

`Frontend con Leaflet`

- Mostrar todos los aeropuertos sobre un mapa (home).
- Usar `Leaflet.markercluster` para agrupar markers.
- Al hacer clic en un marcador:
  - Enviar un `GET /airports/{iata_code}` al backend.
  - Mostrar un popup con los datos reales del aeropuerto.
  - Esta acción **cuenta como una visita** para popularidad.

`Pasos para importar json dentro de un contenedor mongo`

- `docker cp data_trasport.json mongo-heroes:/tmp/data_trasport.json`
- `docker exec -it mongo-heroes mongoimport --db airport_db --collection airports --drop --file /tmp/data_trasport.json`

`Docker Compose`
`Configurar todos los servicios en un archivo docker-compose.yml, incluyendo:`

- `mongo`: base de datos.
- `redis-geo`: para ubicación.

- `redis-pop`: para popularidad.
- `backend`: API y carga de datos.
- `frontend`: visualización Leaflet