Taller : Arquitectura 4-servicios con separación de lecturas y escrituras (R/W) en Docker

Fase 1 — Diseño y orquestación

Meta: dejar lista la topología y el compose.

- Definir variables de entorno: puertos, usuarios, contraseñas, nombres de BD, URLs `PRIMARY` y `REPLICA`.
- Dibujar el diagrama propio con flujos: `Usuario → Frontend → API → (Primary=ESCRITURAS / Replica=LECTURAS)`.
- Preparar estructura de carpetas del proyecto y el archivo de orquestación (servicios: `frontend`, `api`, `db-primary`, `db-replica`), redes internas, volúmenes y healthchecks.

Criterios de éxito: los 4 servicios arrancan y quedan "healthy".
Evidencias: diagrama actualizado + salida de `docker compose ps`/logs.

Fase 2 — Datos y replicación

Meta: base de datos funcional con streaming replication.

- Definir una tabla simple (p. ej., `items` o `todos`) y un script de semilla.
- Configurar primaria y réplica; verificar que la réplica sigue a la primaria.
- Probar inserciones en la primaria y confirmar aparición en la réplica; medir retraso observado.

Criterios de éxito: conteos y datos coinciden tras ponerse al día la réplica.
Evidencias: capturas/consultas mostrando inserción en primaria y lectura en réplica; logs de ambas instancias.

Fase 3 — Backend + Frontend (separación de lecturas y escrituras (R/W))

Meta: API con dos pools (`PRIMARY` para `POST/PUT/DELETE` y `REPLICA` para `GET`) y un frontend mínimo que consuma la API.

- **Backend:** rutas de lectura y escritura; variable para cada pool; endpoint `/health`; logs que indiquen qué pool usó cada petición.
- **Frontend:** página estática con lista (`GET`) y formulario simple (`POST/PUT`); manejo básico de errores.

Criterios de éxito: `GET` se atiende desde réplica; escrituras desde primaria (verificable por logs y por la aparición posterior del dato en lecturas).
Evidencias: capturas del cliente (o frontend) + extractos de logs donde se vea `READ: REPLICA / WRITE: PRIMARY`.

Fase 4 — Validación, resiliencia y sustentación

Meta: demostrar comportamiento correcto y tolerancia a fallas.

Pruebas:

- Flujo normal: crear varios registros y leerlos.
- **Lag de réplica:** medir tiempo hasta que un `POST` aparece en `GET`.
- Caída de réplica: `GET` debe fallar controladamente (mensaje claro).
- Caída de primaria: `GET` sigue funcionando; `POST/PUT/DELETE` fallan controladamente.

Informe corto mínimo de 3 páginas: decisiones, supuestos, resultados de pruebas, lecciones.
Guión de sustentación (5–7 min): arquitectura, separación de lecturas y escrituras (R/W), evidencia de replicación, manejo de fallas.

Criterios de éxito: todas las pruebas reproducidas y documentadas; explicación clara de riesgos (leer contra primaria como plan B, impacto en consistencia).
Evidencias: tabla/resumen de pruebas con resultado esperado vs observado + capturas/logs.

Entregables

- Diagrama propio, archivo de orquestación, estructura del repo.
- Evidencias de replicación y separación de lecturas (GET) hacia la réplica y de escrituras (POST/PUT/DELETE) hacia la primaria.
- Informe técnico + guion de sustentación.

Rúbrica (rápida)

- Infra y orquestación: **25%**
- Replicación verificable: **20%**
- Evidencia clara de separación de lecturas y escrituras (con logs): **35%**
- Validación, resiliencia y claridad en la sustentación: **20%**