Capítulo 4: Gestionar paginación, caché e indización para mejorar rendimiento de consultas

1. Gestionar paginación:

Definición:
La paginación divide un conjunto grande de resultados en páginas manejables. Reduce la cantidad de datos transferidos y la carga del servidor. Es esencial en listados web y APIs para evitar tiempos de respuesta largos y consumo excesivo de memoria en el cliente o servidor.

Problema que resuelve:
Evita respuestas masivas que ralentizan la aplicación y generan timeouts. Limita la carga en la base de datos y en la red. También mejora la experiencia de usuario al mostrar datos por tramos y permite técnicas adicionales como prefetching o cache por página.

Cómo implementarla:

- Usa LIMIT/OFFSET con moderación; para grandes tablas, prefiere paginación por cursor.
- Devuelve metadatos: total, página actual y tamaño por página.
- En la API, soporta parámetros page y size; documenta límites máximos.

Errores comunes y mitigación:
Usar OFFSET grande provoca escaneos lentos en muchas bases de datos. Solución: paginación por cursor (id mayor/menor) o índices adecuados sobre la columna de ordenación. Evita tamaños de página excesivos; 20–100 filas suele ser razonable.

Tip para examen / práctica:
En pruebas te pedirán explicar LIMIT/OFFSET frente a cursor y escribir la consulta paginada. Tú deberías saber implementar ambas y justificar cuándo cambia el rendimiento según el tamaño de la tabla.

Ejemplo Paginación:
Imagina que tienes una tabla usuarios con 1.200.000 filas; paginar con LIMIT 50 y un cursor por id reduce tiempo de respuesta y evita leer páginas anteriores completas.

2. Gestionar caché:

Definición:
El caché guarda resultados frecuentes para servirlos más rápido sin volver a consultar la base de datos. Puede situarse en el cliente, servidor web, proxy o en memoria externa (Redis, Memcached). Es como un atajo que evita trabajo repetido.

Tipos y niveles: