

## Implementación de Cache (con Redis)

Para una aplicación E-commerce personal:

### **1. Tema Asignado:**

El tema asignado para el presente trabajo es la implementación de un sistema de caché en una aplicación web, utilizando Redis como tecnología principal.

La caché es un mecanismo de almacenamiento temporal de datos cuyo objetivo es reducir tiempos de acceso y mejorar el rendimiento general del sistema. En aplicaciones modernas, especialmente aquellas con arquitectura cliente-servidor, es común que múltiples solicitudes accedan repetidamente a la misma información. En estos casos, consultar constantemente la base de datos genera latencia innecesaria y sobrecarga del sistema.

El objetivo de esta Prueba de Concepto (PoC) fue implementar una solución funcional que incorpore una capa de caché real dentro de una arquitectura desacoplada, demostrando comprensión práctica del concepto y su impacto en el rendimiento.

### **2. Problema que intenta resolver:**

Las aplicaciones web modernas trabajan con múltiples peticiones constantemente. En un e-commerce, por ejemplo, es común que los usuarios consulten repetidamente el listado de productos disponibles.

En este proyecto, el frontend realiza solicitudes para obtener un listado de videojuegos desde una API externa (MockAPI), que simula una base de datos remota.

Sin la implementación de caché:

- Cada solicitud implica una conexión remota.
- Se ejecuta nuevamente la consulta.
- Se procesan los datos.
- Se transfiere la información completa.
- Se genera latencia constante.

Esto provoca:

- Mayor tiempo de respuesta.
- Consumo innecesario de recursos.
- Sobrecarga del servidor.
- Experiencia de usuario menos eficiente.

El problema central es la ineficiencia en consultas repetidas a datos que no cambian constantemente.

La implementación de caché permite almacenar temporalmente esos datos en memoria para evitar repetir la consulta original.

### 3. Marco Teórico – ¿Qué es la Caché?

La caché es un mecanismo de almacenamiento temporal de alta velocidad que guarda copias de datos frecuentemente utilizados para reducir el tiempo de acceso.

En arquitectura web, la caché puede implementarse en distintos niveles:

- Caché de navegador
- Caché de servidor
- Caché de base de datos

En este proyecto se implementó caché del lado del servidor, utilizando Redis como sistema de almacenamiento en memoria.

Características principales de la caché:

- Acceso extremadamente rápido (memoria RAM)
- Almacenamiento temporal
- Configuración de tiempo de expiración (TTL – Time To Live)
- Reducción de carga en la base de datos
- Mejora significativa en rendimiento

### 4. Arquitectura de la solución:

Se implementó una arquitectura desacoplada con separación clara de responsabilidades:

1. El frontend solicita los productos al endpoint:  
GET /products
2. El backend recibe la solicitud y consulta Redis:
  - Si la clave "products" existe → responde desde caché.
  - Si no existe → consulta la base externa (MockAPI).
3. Si consulta la base:
  - Obtiene los datos.
  - Los almacena en Redis.
  - Define un TTL de 60 segundos.
  - Devuelve la respuesta al frontend.
4. Las siguientes solicitudes dentro del TTL se responden directamente desde Redis.

Se implementó además un endpoint adicional:

POST /cache/invalidate

Este endpoint permite borrar manualmente la caché para forzar una nueva consulta a la base de datos.

Redis se ejecuta en un contenedor Docker, simulando un entorno profesional de despliegue.

## 5. Tecnologías Utilizadas:

### Frontend:

- React: Biblioteca para construcción de interfaces de usuario.
- React Bootstrap: Framework de componentes visuales, con CSS.
- React Router: Manejo de navegación.
- Context API: Gestión de estado global.
- Vite: Herramienta de desarrollo y bundling.

### Backend:

- Node.js: Entorno de ejecución JavaScript del lado del servidor.
- Express: Framework minimalista para APIs REST.
- Redis: Sistema de almacenamiento en memoria utilizado como caché.
- node-fetch: Para realizar solicitudes HTTP a la API externa.

### Infraestructura:

- Docker: Para ejecutar Redis en contenedor.
- Git y GitHub: Control de versiones y entrega del proyecto.

### Base de datos simulada:

- MockAPI: Servicio externo utilizado para simular una base de datos remota.

## 6. Resultados obtenidos:

Para demostrar la mejora, se simuló una base de datos lenta con un delay artificial de 2 segundos.

Primera solicitud:

- Fuente: DATABASE
- Tiempo promedio: ~2000 ms

Solicitudes posteriores:

- Fuente: CACHE
- Tiempo promedio: 5–30 ms

Se obtuvo una reducción de tiempo superior al 95%.

Esto evidencia el impacto significativo que tiene la implementación de una capa de caché en aplicaciones reales.

## **7. Principales dificultades encontradas:**

Durante el desarrollo surgieron los siguientes desafíos:

- Configuración inicial de Redis en Docker.
- Comprensión del funcionamiento de TTL.
- Integración entre frontend y backend desacoplados.
- Manejo de invalidación manual de caché.

Cada dificultad permitió profundizar la comprensión de arquitecturas reales utilizadas en sistemas productivos.

## **Conclusión:**

La implementación de un sistema de caché utilizando Redis permitió optimizar significativamente el rendimiento de la aplicación.

Se logró:

- Reducir latencia.
- Disminuir carga sobre la base de datos.
- Mejorar experiencia de usuario.
- Aplicar arquitectura profesional desacoplada.
- Implementar infraestructura con contenedores Docker.

Esta Prueba de Concepto demuestra la importancia de las técnicas de optimización en el desarrollo de software moderno y evidencia comprensión tanto teórica como práctica del concepto de caché.