# Tarea — Parte 1 (Informe)

1) ¿Qué es localStorage y para qué se utiliza?

localStorage es parte de la Web Storage API del navegador y permite guardar pares clave–valor de forma persistente en el dispositivo, por origen (dominio+protocolo+puerto). A diferencia de sessionStorage, su información no se elimina al cerrar la pestaña o el navegador (permanece hasta que el usuario o la aplicación la borre). Se usa para recordar preferencias de interfaz, estados de formularios, carritos simples, banderas de “no volver a mostrar”, pequeños cachés de respuestas públicas, entre otros.

2) Diferencias entre localStorage, sessionStorage y cookies

• Duración: localStorage persiste hasta ser eliminado; sessionStorage vive mientras la pestaña/ventana siga abierta; las cookies dependen de su fecha de expiración.

• Alcance: localStorage se comparte entre pestañas de la misma “origen”; sessionStorage es exclusivo de cada pestaña/ventana; las cookies aplican por dominio/ruta y pueden incluirse para subrutas o subdominios según configuración.

• Envío al servidor: localStorage y sessionStorage NO viajan en cada solicitud; las cookies SÍ (a menos que sean solo “JavaScript-only” sin enviar). Por ello, cookies afectan el ancho de banda y son útiles para sesiones del lado servidor.

• Tamaño: localStorage/sessionStorage suelen ofrecer ~5–10 MB por origen (varía por navegador); cada cookie ronda ~4 KB.

• Seguridad y controles: cookies pueden marcarse HttpOnly/Secure/SameSite (lo que restringe acceso desde JS); Web Storage es accesible desde JS, por lo que no debe guardar información sensible. Ante un XSS, localStorage y sessionStorage pueden ser leídos.

• API: Web Storage usa métodos sencillos setItem/getItem/removeItem/clear; cookies usan document.cookie (cadena), requiriendo parseo manual.

• Eventos: cambios en localStorage disparan el evento “storage” en otras pestañas del mismo origen; sessionStorage no se comparte entre pestañas.

3) ¿Qué tipos de datos se pueden almacenar en localStorage?

Internamente todo se guarda como texto (strings). Para objetos, arreglos o números conviene serializar con JSON.stringify() y, al leer, usar JSON.parse(). No se deben guardar funciones, datos binarios sin codificar, ni información sensible (tokens secretos, contraseñas, datos personales).

4) Ventajas y desventajas de usar localStorage

Ventajas

• Sencillo de usar (API mínima y clara).

• Capacidad mayor que cookies y sin enviarse al servidor en cada request.

• Persistencia local y disponible offline.

• Útil para preferencias, borradores y estados de UI.

Desventajas

• Acceso síncrono: abuso de lecturas/escrituras grandes puede bloquear la UI.

• Expuesto a JavaScript: si hay XSS, un atacante podría leerlo. No almacenar datos sensibles.

• Sin expiración automática: si se requiere caducidad, hay que implementarla manualmente.

• Cuotas/limitaciones varían entre navegadores y pueden limpiarse por políticas del usuario/sistema.

• No es una base de datos: sin índices ni consultas; solo pares clave–valor.

5) Ejemplos de uso en aplicaciones modernas

• Preferencias de usuario (tema oscuro/idioma/vistas).

• Recordar filtros y paginaciones.

• Borradores de formularios (autosave) para no perder el progreso al recargar.

• Carritos sencillos (no sensibles) y banderas de onboarding (“no volver a mostrar”).

• Caché de respuestas públicas de APIs, con control manual de caducidad.