	I.E.S. HERMANOS MACHADO **2º CFGS DAW** **Desarrollo Web en Entorno Cliente** **Examen UD04 – 20/02/2026**	**DEPARTAMENTO DE INFORMÁTICA**

Cree un proyecto React+TS llamado "tech-inventory-exam". Una vez creado, copia el contenido de la carpeta "src" del zip al de tu proyecto. A continuación, debes empezar a codificar sin conexión de red. Está prohibido el uso de modelos de inteligencia artificial que ayuden en la codificación de la solución.

El departamento de IT del instituto necesita una pequeña aplicación web para llevar el control del material informático (portátiles, monitores, periféricos, etc.). Se te proporciona una carpeta src para copiar en tu proyecto. En ella ya va el archivo App.css con todos los estilos ya configurados. Tu misión es desarrollar la lógica de la aplicación aplicando los conceptos vistos en clase.

Instrucciones Generales:

- No necesitas escribir CSS, céntrate en la lógica de React y TypeScript.
- Está estrictamente prohibido el uso del tipo any. Debes tipar correctamente todas las *props* y estados.
- Respeta el principio de inmutabilidad al actualizar arrays y objetos en el estado.

Instrucciones de Estilos (CSS): Para que la aplicación luzca correctamente, asegúrate de importar App.css en tu componente principal (App.tsx) y utiliza las clases (className) que se te indican en cada uno de los ejercicios.

Ejercicio 1: Tipado y Componente de Presentación (2,5 puntos)

Modelo de Datos: En el archivo correspondiente (ej. src/types/index.ts), define una interfaz llamada InventoryItem que contenga las siguientes propiedades: id (string), name (string), category (string), quantity (number) e isCritical (boolean, opcional).

Componente InventoryCard: Crea este componente para que reciba por *props* un objeto de tipo InventoryItem. El contenedor principal del componente debe llevar la clase className="card".

Renderizado Condicional en la Tarjeta:

- Si la cantidad (quantity) es 0, el contenedor principal de la tarjeta debe llevar también la clase out-of-stock (ej. className="card out-of-stock"), y en lugar de la cantidad numérica debe mostrar el texto "Agotado" con la clase className="error".

Si el ítem está marcado como crítico (isCritical: true), debe mostrar un con el texto "⚠ Crítico" y la clase className="badge critical" junto al nombre.

Ejercicio 2: Estado e Interactividad (2,5 puntos)

Estado Inicial: En App.tsx, crea un estado para almacenar los ítems del inventario. Inicialízalo con un array que contenga al menos 3 objetos de prueba que cumplan la interfaz InventoryItem.

Listado: Renderiza una lista de componentes InventoryCard recorriendo tu estado de ítems. Envuelve esta lista en un contenedor con la clase className="grid".

Modificar Cantidad: * Añade dos botones en el componente InventoryCard (uno para sumar + y otro para restar -). Envuelve los textos de cantidad y los botones en un con la clase className="quantity-controls".

- Al hacer clic en los botones, deben modificar la cantidad de ese ítem específico en el estado general de la aplicación.
- **Restricción:** La cantidad de un ítem nunca puede ser menor que 0.

Ejercicio 3: Formularios Controlados (2,5 puntos)

Componente AddItemForm: Crea un formulario para añadir nuevo material. La etiqueta <form> debe llevar la clase className="form". Debe tener inputs controlados mediante estado para: Nombre (texto), Categoría (select), Cantidad (número) y Material Crítico (checkbox).

Gestión del Envío (Submit): Al enviar el formulario, el nuevo ítem debe añadirse al estado global del inventario. Genera un id único para el nuevo ítem (puedes usar crypto.randomUUID()).

Limpieza: Tras añadir el ítem con éxito, los campos del formulario deben volver a sus valores iniciales (vacíos o por defecto).

Ejercicio 4: Efectos, Referencias y Custom Hooks (2,5 puntos)

Auto-focus: Utiliza el hook useRef junto con useEffect para que, nada más cargar la página, el cursor se posicione automáticamente en el input "Nombre" del formulario.

Notificación Temporal: Crea un estado en App.tsx para mostrar un mensaje (ej. "¡Ítem añadido!"). Cuando se añada un ítem a través del formulario, muestra este mensaje dentro de un con la clase className="notification".

Limpieza del Timer: Utiliza useEffect y un temporizador para que el mensaje anterior desaparezca automáticamente a los 3 segundos. Recuerda limpiar el temporizador (función *cleanup*) para evitar fugas de memoria.

Refactorización a Custom Hook: Extrae toda la lógica de gestión del inventario (el estado del array, la función de añadir y la función de actualizar cantidad) a un custom hook llamado useInventory. Consume este hook desde App.tsx.

Normas:

- Se debe codificar todo el código en los ficheros indicado en el enunciado de cada ejercicio.
- Entregar un único fichero zip con la carpeta **src** de tu proyecto.