# EXAMEN: Gestión de Tienda Online

**Fecha: __ 07/11/2025___**

## ESTRUCTURA DEL PROYECTO

```
src/
├── db/
│   └── data.js                 ← DATOS INICIALES (ya proporcionado)
├── helpers/
│   └── tiendaOnline.js         ← TÚ IMPLEMENTAS AQUÍ
└── app.js                       ← PRUEBAS (opcional)
```

## EJERCICIO 1: `crearInventario()` (1.5 puntos)

Crear un **Map** con todos los productos de `data.js`. Cada producto debe tener:

- Todas sus propiedades originales
- `alertaStock` = `false` (inicialmente)
- `contadorFavoritos` = cuántas veces aparece el ID del producto en el array `favoritos` de `data.js`

Guardar en LocalStorage con clave `"inventario"`. Devolver el Map.

### Ejemplo:

```
const inventario = crearInventario();
console.log(inventario.size); // 8
console.log(inventario.get(101).contadorFavoritos); // 3
console.log(inventario.get(108).contadorFavoritos); // 0
```

## EJERCICIO 2: `actualizarStock(idProducto, cantidad)` (1.5 puntos)

Actualizar el stock de un producto y activar alerta si cae bajo 10 unidades.

**Validaciones:**

- Si el producto no existe → `throw new Error("Producto no encontrado")`
- Calcular `nuevoStock = stock + cantidad`
- Si `nuevoStock < 10` → `alertaStock = true`, si no → `false`
- Guardar cambios en LocalStorage (clave: `"inventario"`)
- Devolver el producto actualizado

### Ejemplo:

```
actualizarStock(101, -15); // stock: 25 - 15 = 10
// { id: 101, stock: 10, alertaStock: false, ... }

actualizarStock(101, -5); // stock: 10 - 5 = 5
// { id: 101, stock: 5, alertaStock: true, ... }

actualizarStock(999, -5); // Error: Producto no encontrado
```

## EJERCICIO 3: `generarInformeStock()` (1.5 puntos)

Devolver objeto con 3 estadísticas del inventario:

- `totalProductos`: cantidad total (8)
- `productosConAlerta`: cuántos tienen `alertaStock = true`
- `valorTotalInventario`: suma de `stock × precio` para cada producto

### Ejemplo:

```
const informe = generarInformeStock();
// {
//   totalProductos: 8,
//   productosConAlerta: 2,
//   valorTotalInventario: 10542.34
// }
```

## EJERCICIO 4: `buscarProductos(nombre)` (1.5 puntos)

Buscar productos por nombre (case-insensitive, búsqueda parcial).

**Pasos:**

1. Convertir `nombre` a minúsculas
2. Filtrar productos cuyo nombre contenga el texto
3. Ordenar por `valoracion` descendente
4. Devolver array

### Ejemplo:

```
buscarProductos("gaming");
// [
//   { id: 101, nombre: "Laptop Gaming Pro", valoracion: 4.5, ... },
//   { id: 104, nombre: "Teclado Mecanico RGB", valoracion: 4.4, ... },
//   { id: 106, nombre: "Raton Gaming", valoracion: 4.3, ... }
// ]

buscarProductos("xyz");
// []
```

## EJERCICIO 5: `obtenerFavoritos()` (0.75 puntos)

Recuperar y devolver los favoritos guardados en LocalStorage.

**Clave:** `"favoritos"`

Si no existe, devolver array vacío `[]`.

## EJERCICIO 6: `guardarFavorito(idProducto)` (0.75 puntos)

Agregar un producto a favoritos (sin duplicados).

**Pasos:**

1. Recuperar favoritos actuales
2. Si el ID ya existe, no hacer nada
3. Si NO existe, agregarlo
4. Guardar en LocalStorage (clave: `"favoritos"`)

## EJERCICIO 7: `agregarAlCarrito(idUsuario, idProducto, cantidad)` (2 puntos)

Agregar un producto al carrito de un usuario.

**Clave en localStorage:** `"carrito_" + idUsuario` (ej: `"carrito_1"`)

**Validaciones:**

- Producto existe en inventario → si no: `throw new Error("Producto no encontrado")`
- Stock suficiente → si no: `throw new Error("Stock insuficiente")`

**Lógica:**

- Si el producto ya está en el carrito: sumar cantidad
- Si NO está: crear nuevo item con `{ id, nombre, precio, cantidad }`

**Item en carrito:** `{ id, nombre, precio, cantidad }`

Guardar cambios en LocalStorage. No devuelve nada.

### Ejemplo:

```
agregarAlCarrito(1, 101, 1); // Agrega 1 Laptop
agregarAlCarrito(1, 104, 2); // Agrega 2 Teclados

agregarAlCarrito(1, 999, 1); // Error: Producto no encontrado
agregarAlCarrito(1, 106, 1000); // Error: Stock insuficiente
```

## EJERCICIO 8: `obtenerCarrito(idUsuario)` (1 punto)

Recuperar todos los productos en el carrito de un usuario.

Si no existe carrito, devolver array vacío `[]`.

Devolver array de productos: `{ id, nombre, precio, cantidad }`

## EJERCICIO 9: `calcularTotalCarrito(idUsuario)` (1.5 puntos)

Calcular el total del carrito usando **obligatoriamente** `reduce()`.

Suma de `(precio × cantidad)` para cada producto.

### Estructura de reduce:

```
array.reduce((acumulador, elemento) => {
  return acumulador + elemento.precio * elemento.cantidad;
}, 0);
```

## EJERCICIO 10: `generarInformeCompleto()` (0.75 puntos)

Integrar datos de varios ejercicios anteriores.

Devolver objeto con:

- `totalProductos`
- `productosConAlerta`
- `valorTotalInventario`
- `totalFavoritos`
- `totalUsuarios`
- `totalPedidos`

## 📊 RESUMEN

| # | Función | Puntos | Dificultad |
|---|---|---|---|
| 1 | `crearInventario()` | 1.5 | ⭐⭐⭐ |
| 2 | `actualizarStock()` | 1.5 | ⭐⭐⭐ |
| 3 | `generarInformeStock()` | 1.5 | ⭐⭐ |
| 4 | `buscarProductos()` | 1.5 | ⭐⭐ |
| 5 | `obtenerFavoritos()` | 0.75 | ⭐ |
| 6 | `guardarFavorito()` | 0.75 | ⭐ |
| 7 | `agregarAlCarrito()` | 2.0 | ⭐⭐⭐⭐ |
| 8 | `obtenerCarrito()` | 1.0 | ⭐ |
| 9 | `calcularTotalCarrito()` | 1.5 | ⭐⭐ |
| 10 | `generarInformeCompleto()` | 0.75 | ⭐ |
| | **TOTAL** | **10.0** | |

## ⚙ TIPS IMPORTANTES

1. **Documenta al menos 2 funciones de tu examen :**
2. **Prueba tu examen a través de app.js**
3. **Añade comentarios siempre que sea posible**

4. **Todos los ejercicios de código deben de tener arriba la cabecera con //@autor: TU NOMBRE**