GUÍA TEÓRICO-PRÁCTICA:

UNIDAD I

SEMANA 7: EVALUACIÓN PRÁCTICA

TEMA 7.1: EVALUACIÓN PRÁCTICA

Proyecto final integrador.

🎯 RETOS FINALES DE LA UNIDAD 1

RETO 1: "SISTEMA DE GESTIÓN DE BIBLIOTECA"

Objetivo: Crear una aplicación completa para gestionar una biblioteca usando todos los conceptos aprendidos.

Requisitos funcionales:

Parte 1 - Estructura de datos:

1. **Crea un array de objetos libros con propiedades:**
    - **id (único)**
    - **titulo**
    - **autor**
    - **anio**
    - **genero (ficcion, ciencia, historia, otros)**
    - **disponible (boolean)**
    - **vecesPrestadas (contador)**

Parte 2 - Funcionalidades: 2. Implementa las siguientes funciones:

- **agregarLibro(titulo, autor, anio, genero): Agrega un nuevo libro**
- buscarLibro(titulo): Busca un libro por título (parcial o total)
- prestarLibro(id): Marca el libro como no disponible e incrementa vecesPrestadas
- devolverLibro(id): Marca el libro como disponible
- eliminarLibro(id): Elimina un libro del array

- getEstadisticas(): Retorna objeto con:
    - totalLibros
    - librosDisponibles
    - librosPrestados
    - libroMasPrestado

Parte 3 - Interfaz de usuario: 3. Crea un HTML con:

- Formulario para agregar libros (con validación)
- Campo de búsqueda en tiempo real (onkeyup)
- Lista de libros mostrando:
    - Título, autor, año
    - Estado (disponible/prestado) con color diferente
    - Botones: Prestar, Devolver, Eliminar
- Sección de estadísticas que se actualice automáticamente

Parte 4 - Características avanzadas: 4. Agrega:

- **Validación de formulario completa**
- **Confirmación antes de eliminar**
- **Filtro por género (select onchange)**
- **Ordenamiento (por título, año, veces prestadas)**
- **Persistencia en localStorage (opcional)**
- **Animaciones CSS al agregar/eliminar**
- **Diseño responsive**

Entregables:

- Código HTML completo
- Código JavaScript documentado con JSDoc
- Capturas de pantalla de la aplicación funcionando
- README explicando cómo usar la aplicación

Criterios de evaluación: ✓ Uso correcto de arrays y objetos (20%) ✓ Manipulación del DOM (20%) ✓ Eventos y validaciones (20%) ✓ Funciones y lógica de programación (20%) ✓ Diseño y experiencia de usuario (20%)

RETO 2: "APLICACIÓN DE CLIMA CON INTERFAZ DINÁMICA"

Objetivo: Crear una aplicación interactiva que simule un sistema de consulta del clima con interfaz dinámica.

Requisitos funcionales:

Parte 1 - Estructura de datos:

1. Crea un objeto ciudades que contenga:
    - Nombre de la ciudad
    - Temperatura (número aleatorio entre -10 y 40)
    - Condición (soleado, nublado, lluvioso, nevando)
    - Humedad (porcentaje)
    - Viento (km/h)
    - Historial (array de temperaturas de los últimos 7 días)

Parte 2 - Funcionalidades: 2. Implementa:

- agregarCiudad(nombre): Agrega ciudad con datos aleatorios
- actualizarClima(ciudad): Actualiza temperatura y condición aleatoriamente
- getPromedioTemperatura(ciudad): Calcula promedio del historial
- getCiudadMasCalida(): Retorna la ciudad con mayor temperatura
- filtrarPorCondicion(condicion): Retorna ciudades con esa condición
- generarRecomendacion(ciudad): Retorna recomendación según clima:
    - Si hace calor (>30): "Usa ropa ligera y hidrátate"
    - Si hace frío (<10): "Abrígate bien"

    - Si llueve: "Lleva paraguas"
    - Si nieva: "Precaución en las calles"

Parte 3 - Interfaz interactiva: 3. Crea:

- Selector de ciudad (dropdown)
- Visualización del clima actual con:
    - Icono según condición (usa emojis o imágenes)
    - Temperatura grande y destacada
    - Detalles (humedad, viento)
    - Color de fondo que cambie según:
        - Soleado: degradado amarillo/naranja
        - Nublado: gris
        - Lluvioso: azul oscuro
        - Nevando: blanco/azul claro
- Gráfico simple del historial (barras con divs)
- Recomendación dinámica

Parte 4 - Interactividad avanzada: 4. Agrega:

- Actualización automática cada 10 segundos (setInterval)
- Animación de transición entre ciudades
- Búsqueda predictiva (autocomplete)
- Modo oscuro/claro (toggle)
- Exportar datos a JSON (descargar archivo)
- Teclas de acceso rápido:
    - Flechas arriba/abajo: cambiar ciudad
    - "R": refrescar clima
    - "N": nueva ciudad
- Validación de entrada

- Mensajes de error personalizados

Parte 5 - Objetos y métodos avanzados: 5. Implementa:

- Constructor o clase para Ciudades
- Métodos encadenados (method chaining)
- Uso de this correctamente
- Closures para contadores privados
- Callbacks o promesas (simuladas)

Entregables:

- Código completo y documentado
- Diagrama de flujo de la aplicación
- Video demo de 2 minutos mostrando funcionalidades
- Archivo README.md con:
    - Instrucciones de instalación
    - Lista de características
    - Capturas de pantalla
    - Lecciones aprendidas

Criterios de evaluación: ✓ Estructura de datos y objetos (15%) ✓ Funciones y algoritmos (20%) ✓ Manipulación del DOM y eventos (20%) ✓ CSS dinámico y animaciones (15%) ✓ Validaciones y manejo de errores (15%) ✓ Creatividad y funcionalidades extra (15%)

Bonus points (+10%):

- **Usar API real del clima (OpenWeatherMap)**
- **Geolocalización del usuario**
- **Gráficos con Chart.js**
- **PWA (Progressive Web App)**