Actividad evaluable — “Gestión de Artistas (DOM + Objetos + Formularios + Tablas)”

Objetivo

Construir una mini-app en una sola página para **crear, listar, buscar, ordenar, editar y borrar** artistas en memoria. Debe usar:

- **DOM**: `querySelector`, manejo de eventos y modificación del árbol.
- **Objetos**: un modelo `Artista` (clase o función constructora) y una colección en memoria.
- **Formularios**: validación con atributos HTML5 y feedback accesible.
- **Tablas**: renderizado semántico de datos con `<caption>`, `<thead>`, `<tbody>`.

> Accesibilidad mínima: etiquetas `<label for>`, `aria-live` para mensajes y tabla con encabezados correctos; navegación por teclado sin trampas.

Enunciado (lo que tienen que hacer)

1. **Modelo de datos**
   Implementa un objeto `Artista` con propiedades: `id`, `nombre`, `disciplina`, `pais`, `anioNacimiento`, `puntuacion` (0–10), `activo` (boolean). La lista viva de artistas será un array en memoria.

2. **Formulario**
   Crea un **formulario** para alta/edición con validación nativa (required, rangos, tipos) y feedback en un `role="status" aria-live="polite"`. Botones **Guardar** (crear/actualizar) y **Cancelar**.

3. **Tabla**
   Muestra la lista en una **tabla HTML** con `<caption>`, `<thead>` y `<tbody>`. Incluye botones por fila: **Editar** y **Borrar**. Los `<th>` del encabezado deben poder **ordenar** por columna al hacer clic.

4. **Búsqueda / filtro**
   Input de búsqueda que filtre por nombre, país o disciplina en tiempo real (evento `input`) o al pulsar en un botón.

5. **Eventos y DOM**
   Usa **addEventListener** para manejar `submit`, `click` (acciones de fila), `input` (filtro) y actualiza el DOM sin recargar.

Buenas prácticas: separa HTML/JS/CSS (enlaza `app.js` y `styles.css`).

Entregables

- `index.html` (estructura + formulario + tabla + plantilla de fila)
- `app.js` (lógica DOM/objetos/eventos)
- `styles.css` (estilos opcionales)

Rúbrica (10 puntos)

Criterio	0–4 Insuficiente	5–6 Suficiente	7–8 Notable	9–10 Sobresaliente	Peso
DOM & Eventos	No usa listeners o manipulación básica incorrecta.	Listeners básicos (submit/click) y render mínimo.	Render dinámico fluido; filtro/orden funcionan en parte.	CRUD completo, filtro en vivo, orden estable y sin recargas; DOM limpio y eficiente.	**30%**
Modelo de Objetos	Sin objeto `Artista` o datos sueltos.	Objeto simple sin validación.	Objeto bien definido y colección coherente.	Encapsulación clara, ids únicos, utilidades (comparadores, serialización).	**20%**
Formularios & Validación (A11y)	Sin `label`, sin required/tipos.	Validación mínima (required) y algunos labels.	Validación HTML5 correcta, mensajes básicos.	Labels + descripciones, `aria-live` para avisos, foco gestionado; tipos/rangos/patrones bien usados. turn1file12	**20%**
Tabla & Presentación Semántica	Lista sin tabla o tabla no semántica.	Tabla con `<thead>/<tbody>` pero uso limitado.	Estructura completa con `caption` y `scope`.	Semántica impecable, accesible por teclado y clara jerarquía de encabezados.	**20%**

Calidad del código	Desorden, sin comentarios.	Algo de formato.	Código legible y organizado.	Módulos/funciones pequeñas, nombres expresivos y comentarios útiles.	**10%**

Pistas de accesibilidad exigidas por la rúbrica: navegación por teclado, etiquetas adecuadas, contraste adecuado (si hay CSS) y mensajes comprensibles.

Sugerencias de implementación en `app.js` (guía rápida)

- `const $ = (sel, ctx=document) => ctx.querySelector(sel);` y `$$ = sel => [...document.querySelectorAll(sel)];`
- Clase `Artista` + función `renderTabla()` que clona `#tplFilaArtista` y rellena `data-field`.
- Listeners:
  - `formArtista.addEventListener('submit', onSubmit)` (crear/editar)
  - `tbodyArtistas.addEventListener('click', onAccionFila)` (delegación)
  - `buscar.addEventListener('input', onFiltrar)`
  - `document.querySelectorAll('.sortable').forEach(th => th.addEventListener('click', onOrdenar))`

- `aria-live`: escribe en `#avisos.textContent` los mensajes de éxito/error.

Notas docentes (por si lo pides para los alumnos):

- Eventos y manejo de `addEventListener` + `this.value` + `event.target`: ver unidad de **Eventos / Controles**.
- Selección y cambio de estilos/clases con `querySelector`, `classList`, `setProperty`: útil para destacar orden activo.
- Semántica de tablas/estructura HTML5: repaso de **UF1**.