DWC\WebStorage\criminales\criminalesComentados.js

```
// Se abre o crea la base de datos IndexedDB con nombre "CriminalDB" y versión 1
let db;
const request = indexedDB.open("CriminalDB", 1);

// Evento que se ejecuta si la base de datos necesita ser creada o actualizada
request.onupgradeneeded = (event) => {
    db = event.target.result;

    // Se verifica si el almacén de objetos "criminales" no existe para crearlo
    if (!db.objectStoreNames.contains("criminales")) {
        const store = db.createObjectStore("criminales", { keyPath: "id", autoIncrement: true });

        // Se crean índices para facilitar la búsqueda por nombre y fecha
        store.createIndex("criminal", "nombre", { unique: false });
        store.createIndex("fecha", "fecha", { unique: false });
        /*
        store.createIndex(nombreIndice, clave, opciones)
        "criminal" y "fecha" → Son los nombres de los índices.
        "nombre" y "fecha" → Son las propiedades de los objetos que se indexarán.
        { unique: false } → Permite que varios registros tengan el mismo valor en estos índices.
        ¿Para qué sirven?
        "criminal": Permite buscar criminales por su nombre más rápido.
        "fecha": Permite buscar registros por fecha de detención sin recorrer toda la base de datos.
        Ambos índices mejoran la eficiencia al recuperar datos sin necesidad de hacer un barrido completo de los registros. 🚀
        */
    }
};

// Manejo de evento cuando la base de datos se abre exitosamente
request.onsuccess = (event) => {
    db = event.target.result;
    /*
    event.target: Hace referencia al evento request.onsuccess o request.onupgradeneeded,
    donde se obtiene el resultado de abrir la base de datos.
```

```
    .result: Contiene la instancia de la base de datos IndexedDB que se abrió o creó exitosamente.
    db = ...: Guarda la referencia de la base de datos en la variable global db, permitiendo acceder
    a la base de datos en otras partes del código.
    En resumen, esta línea asigna la base de datos IndexedDB abierta a la variable db para que pueda
    ser utilizada en operaciones futuras, como transacciones o consultas.
    */
    console.log("Conexión a la base de datos establecida.");
};

// Manejo de errores en caso de que la base de datos no pueda abrirse
request.onerror = () => {
    console.error("Error al abrir la base de datos.");
};

// Evento que escucha el clic en el botón de registro de criminales
document.getElementById('registrarBtn').addEventListener('click', registrarCriminal);

// Función para registrar un criminal en la base de datos
function registrarCriminal() {
    // Obtención de valores de los campos del formulario
    const nombre = document.getElementById('nombre').value.trim();
    const fecha = document.getElementById('fecha').value;
    const foto = document.getElementById('foto').files[0];
    const informe = document.getElementById('informe').files[0];

    // Validación para asegurarse de que los campos obligatorios no estén vacíos
    if (!nombre || !foto || !informe) {
        alert("Completa todos los campos.");
        return;
    }

    // Creación de objetos FileReader para leer los archivos adjuntos
    const readerFoto = new FileReader();
    const readerInforme = new FileReader();

    // Lectura del archivo de imagen y conversión a base64
    readerFoto.onload = () => {
```

```
        const fotoUrl = readerFoto.result;

        // Lectura del archivo de informe y conversión a base64
        readerInforme.onload = () => {
            const informeData = readerInforme.result;

            // Se crea el objeto con los datos del criminal
            const registro = { nombre, fecha, foto: fotoUrl, informe: informeData };

            // Se inicia una transacción para escribir en la base de datos
            /*
            db.transaction(["criminales"], "readwrite")
            Crea una transacción sobre el almacén de objetos "criminales".
            Se usa el modo "readwrite" para permitir lectura y escritura (modificación de datos).
            La transacción agrupa múltiples operaciones para garantizar su atomicidad (si una falla, todas se revierten).
            transaction.objectStore("criminales")

            Obtiene el almacén de objetos "criminales" dentro de la transacción.
            Permite realizar operaciones como agregar (add), obtener (get), actualizar (put) o eliminar (delete) registros.
            En resumen, este código inicia una transacción y accede al almacén de objetos "criminales" para operar sobre la
            base de datos IndexedDB.
            */
            const transaction = db.transaction(["criminales"], "readwrite");
            const store = transaction.objectStore("criminales");

            // Se añade el nuevo registro a la base de datos
            store.add(registro);

            // Al completarse la transacción, se muestra un mensaje y se reinicia el formulario
            transaction.oncomplete = () => {
                alert("Criminal registrado.");
                document.getElementById('criminalForm').reset();
                document.getElementById('vistaPreviaFoto').style.display = 'none';
                document.getElementById('nombreFoto').textContent = 'Ningún archivo seleccionado';
                document.getElementById('nombreInforme').textContent = 'Ningún archivo seleccionado';
            };
        };
```

```
109
110         // Se lee el archivo de informe como base64
111         readerInforme.readAsDataURL(informe);
112     };
113
114     // Se lee el archivo de foto como base64
115     readerFoto.readAsDataURL(foto);
116 }
117
118 // Función para mostrar la lista de criminales almacenados en la base de datos
119 function mostrarCriminales() {
120     const resultadosDiv = document.getElementById('resultados');
121     resultadosDiv.innerHTML = '';
122
123     // Se inicia una transacción en modo lectura para obtener los datos
124
125     /*
126     db.transaction(["criminales"], "readonly")
127
128     Inicia una transacción en modo "readonly", lo que significa que solo permite leer datos, sin modificar ni eliminar registros.
129     La transacción se realiza sobre el almacén de objetos "criminales".
130     transaction.objectStore("criminales")
131
132     Accede al almacén de objetos "criminales" dentro de la transacción.
133     Este almacén contiene la información registrada en la base de datos.
134     store.getAll()
135
136     Recupera todos los registros almacenados en el almacén de objetos "criminales".
137     Se obtiene un objeto IDBRequest, cuya propiedad onsuccess debe manejarse para acceder a los datos
138     una vez que la operación se complete correctamente.
139     En resumen, este código se usa para leer todos los criminales registrados en la base de datos sin realizar modificaciones.
140     */
141     const transaction = db.transaction(["criminales"], "readonly");
142     const store = transaction.objectStore("criminales");
143     const consulta = store.getAll();
144
145     // Cuando la consulta es exitosa, se procesan los datos obtenidos
```

```
    consulta.onsuccess = () => {
        const criminales = consulta.result;

        // Si no hay registros, se muestra un mensaje indicando que la base de datos está vacía
        if (criminales.length === 0) {
            resultadosDiv.innerHTML = '<p>No hay criminales registrados.</p>';
        } else {
            // Se recorre la lista de criminales y se muestran sus datos
            criminales.forEach(c => {
                resultadosDiv.innerHTML += `<p><strong>Nombre:</strong> ${c.nombre}, <strong>Fecha de detención:</strong> ${c.fecha
                    || 'En libertad'}</p>`;
            });
        }
    };
}

// Función para buscar criminales por nombre utilizando el índice "criminal"
function buscarPorNombre() {
    const nombre = document.getElementById('buscar-nombre').value.trim();
    const resultadosDiv = document.getElementById('resultados');
    resultadosDiv.innerHTML = '';

    // Se inicia una transacción en modo lectura
    const transaction = db.transaction(["criminales"], "readonly");
    const store = transaction.objectStore("criminales");
    const index = store.index("criminal");
    const consulta = index.getAll();

    // Cuando la consulta es exitosa, se filtra el resultado por nombre
    consulta.onsuccess = () => {
        const criminal = consulta.result.find(c => c.nombre.toLowerCase() === nombre.toLowerCase());
        /*
        consulta.result

        consulta es el objeto IDBRequest que obtiene todos los datos almacenados en el índice "criminal" dentro de IndexedDB.
        .result contiene un array con todos los registros obtenidos.
        .find(c => c.nombre.toLowerCase() === nombre.toLowerCase())
```

```
183
184         Se usa el método .find() para buscar en el array el primer elemento (c) cuyo nombre coincida con el valor
185         de nombre ingresado por el usuario.
186         .toLowerCase() convierte ambos nombres a minúsculas para hacer la comparación insensible a mayúsculas y minúsculas.
187         En resumen:
188         Esta línea busca dentro de la base de datos si existe un criminal con el mismo nombre ingresado en la búsqueda.
189         Si lo encuentra, criminal contendrá el objeto con los datos del criminal; si no, su valor será undefined.
190         */
191
192         // Se muestra el resultado si se encontró un registro, de lo contrario, se indica que no hay coincidencias
193         if (!criminal) {
194             resultadosDiv.innerHTML = `<p>No se encontró ningún criminal con el nombre "${nombre}".</p>`;
195         } else {
196             resultadosDiv.innerHTML = `<p><strong>Nombre:</strong> ${criminal.nombre}, <strong>Fecha de detención:</strong>
    ${criminal.fecha || 'En libertad'}</p>`;
197         }
198     };
199 }
200
```