clientb

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ClienteB - Reproducción</title>
  <style>
    body, html {
      margin: 0;
      padding: 0;
      width: 100%;
      height: 100%;
      background-color: black;
      color: white;
      font-family: Arial, sans-serif;
      overflow: hidden;
    }
    #mediaContainer {
      position: relative;
      width: 100vw;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      overflow: hidden;
    }
    .mediaItem {
      position: absolute;
      width: 100%;
      height: 100%;
      display: flex;
      justify-content: center;
      align-items: center;
      opacity: 0;
      transition: opacity 1s ease-in-out;
    }
    .mediaItem img, .mediaItem video {
      object-fit: cover;
```

```
    width: 100%;
    height: 100%;
  }
  .vertical-pair {
    display: flex;
    width: 100%;
    height: 100%;
  }
  .vertical-pair .mediaItem {
    position: relative;
    width: 50%;
    height: 100%;
    overflow: hidden;
  }
  .vertical-pair .mediaItem img {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: auto;
    animation: moveVertical 16s linear;
  }
  @keyframes moveVertical {
    0% { transform: translateY(0); }
    100% { transform: translateY(calc(-100% + 100vh)); }
  }
  #clock {
    position: fixed;
    bottom: 20px;
    left: 50%;
    transform: translateX(-50%);
    z-index: 2000;
    color: white;
    font-family: Arial, sans-serif;
    text-align: center;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
  }
  #clock .time {
    font-size: 80px;
    font-weight: bold;
```

```
    }
    #clock .date {
      font-size: 24px;
    }
  </style>
</head>
<body>
  <div id="mediaContainer"></div>
  <div id="clock">
    <div class="time" aria-live="polite"></div>
    <div class="date" aria-live="polite"></div>
  </div>
  <script>
    let mediaFiles = [];
    let currentIndex = 0;

    function fetchSelectedPhotos() {
      const stored = localStorage.getItem('selectedPhotos');
      if (stored) {
        mediaFiles = JSON.parse(stored);
        console.log('Medios cargados:', mediaFiles);
      }
    }

    function createMediaElement(fileObj) {
      const mediaItem = document.createElement('div');
      mediaItem.classList.add('mediaItem');
      const file = fileObj.url;
      const isVideo = ['mp4', 'avi', 'mov'].some(ext =>
file.toLowerCase().endsWith(`.${ext}`));

      if (isVideo) {
        const video = document.createElement('video');
        video.src = file;
        video.autoplay = true;
        video.muted = true;
        video.loop = false;
        video.playsInline = true;
        video.controls = false;  // se ocultan controles para que no
aparezca el ícono de "play"
```

```
        mediaItem.appendChild(video);
        // Cuando el video esté listo para reproducir, se lanza play()
        video.addEventListener('canplay', () => {
          video.play().catch(console.error);
        });
        // Se asigna un listener para cambiar la opacidad al finalizar
        video.addEventListener('ended', () => {
          mediaItem.style.opacity = 0;
        });
        // Se muestra el elemento inmediatamente
        mediaItem.style.opacity = 1;
      } else {
        const img = new Image();
        img.src = file;
        img.onload = () => {
          mediaItem.appendChild(img);
          mediaItem.style.opacity = 1;
        };
        img.onerror = () => console.error('Error carga imagen:', file);
      }
      return mediaItem;
    }

    async function getOrientation(fileObj) {
      if (!fileObj?.url) return 'horizontal';
      try {
        const img = await new Promise((resolve, reject) => {
          const image = new Image();
          image.onload = () => resolve(image);
          image.onerror = reject;
          // Se usa la URL original para obtener las dimensiones reales
          image.src = fileObj.url;
        });
        return img.naturalHeight > img.naturalWidth ? 'vertical' :
'horizontal';
      } catch {
        return 'horizontal';
      }
    }
```

```
    async function renderVerticalPair(fileObj1, fileObj2) {
      const mediaContainer = document.getElementById('mediaContainer');
      mediaContainer.innerHTML = '';

      const pairContainer = document.createElement('div');
      pairContainer.classList.add('vertical-pair');

      const item1 = createMediaElement(fileObj1);
      const item2 = createMediaElement(fileObj2);

      pairContainer.appendChild(item1);
      pairContainer.appendChild(item2);
      mediaContainer.appendChild(pairContainer);

      return new Promise(resolve => setTimeout(resolve, 25000));
    }

    async function playMediaLoop() {
      while (true) {
        // Actualiza la lista de medios en cada iteración para reflejar
cambios en localStorage
        fetchSelectedPhotos();

        if (!mediaFiles.length) {
          await new Promise(r => setTimeout(r, 2000));
          continue;
        }

        mediaFiles = mediaFiles.filter(f => f?.url?.startsWith('http'));

        if (currentIndex >= mediaFiles.length) currentIndex = 0;
        const fileObj = mediaFiles[currentIndex];

        if (!fileObj?.url) {
          currentIndex++;
          continue;
        }

        const orientation = await getOrientation(fileObj);
        let delay = 8000;
```

```
        if (orientation === 'vertical' && currentIndex + 1 <
mediaFiles.length) {
          const nextFile = mediaFiles[currentIndex + 1];
          if (nextFile?.url && await getOrientation(nextFile) ===
'vertical') {
            await renderVerticalPair(fileObj, nextFile);
            currentIndex += 2;
            continue;
          }
        }

        const mediaItem = createMediaElement(fileObj);
        const container = document.getElementById('mediaContainer');
        container.innerHTML = '';
        container.appendChild(mediaItem);

        const isVideo = ['mp4', 'avi', 'mov'].some(ext =>
fileObj.url.toLowerCase().endsWith(`.${ext}`));
        if (isVideo) {
          // Se espera a que el video termine de reproducirse
          const video = mediaItem.querySelector('video');
          await new Promise(r => video.addEventListener('ended', r, {
once: true }));
        } else {
          await new Promise(r => setTimeout(r, delay));
        }

        mediaItem.style.opacity = 0;
        await new Promise(r => setTimeout(r, 1000));
        mediaItem.remove();
        currentIndex++;
      }
    }

    function removeMedia(id) {
      mediaFiles = mediaFiles.filter(item => item.id !== id);
      localStorage.setItem('selectedPhotos', JSON.stringify(mediaFiles));
      currentIndex = 0;
    }
```

```
    function updateClock() {
      const now = new Date();
      document.querySelector('#clock .time').textContent =
        now.toLocaleTimeString('es-ES', { hour: '2-digit', minute:
'2-digit' });
      document.querySelector('#clock .date').textContent =
        now.toLocaleDateString('es-ES', { weekday: 'long', year:
'numeric', month: 'long', day: 'numeric' });
    }

    window.onload = () => {
      fetchSelectedPhotos();
      updateClock();
      setInterval(updateClock, 1000);
      playMediaLoop();
      document.addEventListener('dblclick', () => {
        if (!document.fullscreenElement)
document.documentElement.requestFullscreen();
      });
    };
  </script>
</body>
</html>
```

index

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Google Photos App - Configuración</title>
  <!-- Carga de las librerías de Google API -->
  <script src="https://apis.google.com/js/api.js"></script>
  <script src="https://accounts.google.com/gsi/client" async
defer></script>
```

```
<style>
  /* Estilos generales */
  body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f0f2f5;
  }
  /* Header de administración (oculto hasta autenticarse) */
  .fixed-header {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    background: white;
    padding: 15px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    z-index: 1000;
    display: none;
    flex-wrap: wrap;
    gap: 10px;
    align-items: center;
  }
  .button {
    padding: 10px 20px;
    border: none;
    border-radius: 20px;
    cursor: pointer;
    background: #007bff;
    color: white;
  }
  /* Botón extra para gestionar reproducción */
  #manageReproductionButton {
    background: #ffc107;
    color: black;
  }
  #progressBar {
    height: 5px;
    background: #007bff;
    width: 0%;
```

```
    transition: width 0.3s;
    flex-grow: 1;
    margin-top: 10px;
  }
  #photosContainer {
    margin-top: 120px;
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
    gap: 10px;
    padding: 10px;
  }
  #photosContainer img {
    width: 100%;
    height: 150px;
    object-fit: cover;
    border-radius: 8px;
    cursor: pointer;
    transition: transform 0.2s, border 0.2s;
  }
  #photosContainer img[selected] {
    border: 3px solid #007bff;
    transform: scale(0.95);
  }
  #message {
    position: fixed;
    bottom: 10px;
    left: 50%;
    transform: translateX(-50%);
    background: rgba(0,0,0,0.7);
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    font-size: 14px;
    z-index: 2000;
  }
  /* Overlay de bienvenida */
  #welcomeOverlay {
    position: fixed;
    top: 0;
    left: 0;
```

```
    width: 100vw;
    height: 100vh;
    background: rgba(0,0,0,0.85);
    color: white;
    z-index: 3000;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;
    padding: 20px;
  }
  #welcomeOverlay h1 {
    margin-bottom: 20px;
    font-size: 32px;
  }
  #welcomeOverlay p {
    margin-bottom: 30px;
    font-size: 18px;
  }
  #welcomeOverlay .welcomeButton {
    padding: 15px 25px;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    background: #28a745;
    color: white;
    font-size: 18px;
  }
  /* Overlay de elección de modo de uso */
  #usageScreen {
    position: fixed;
    top: 0;
    left: 0;
    width: 100vw;
    height: 100vh;
    background: rgba(0,0,0,0.85);
    color: white;
    z-index: 3000;
    display: none;
```

```
      flex-direction: column;
      justify-content: center;
      align-items: center;
      text-align: center;
      padding: 20px;
    }
    #usageScreen h2 {
      margin-bottom: 30px;
    }
    #usageScreen .usageButton {
      padding: 15px 25px;
      margin: 10px;
      border: none;
      border-radius: 10px;
      cursor: pointer;
      background: #28a745;
      color: white;
      font-size: 16px;
    }
    #usageScreen .usageButton.secondary {
      background: #17a2b8;
    }
    /* Overlay para gestionar reproducción (mostrar imágenes
seleccionadas) */
    #manageOverlay {
      position: fixed;
      top: 0;
      left: 0;
      width: 100vw;
      height: 100vh;
      background: rgba(0,0,0,0.9);
      color: white;
      z-index: 3500;
      display: none;
      flex-direction: column;
      padding: 20px;
      overflow-y: auto;
    }
    #manageOverlay h2 {
      text-align: center;
```

```
    margin-bottom: 20px;
  }
  #manageOverlay .grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
    gap: 10px;
  }
  #manageOverlay .grid img {
    width: 100%;
    height: 150px;
    object-fit: cover;
    border-radius: 8px;
    position: relative;
  }
  #manageOverlay .closeButton {
    align-self: center;
    margin-top: 20px;
    padding: 10px 20px;
    border: none;
    border-radius: 10px;
    background: #dc3545;
    color: white;
    cursor: pointer;
  }
  /* Botón de eliminar en cada imagen */
  .deleteIcon {
    position: absolute;
    top: 5px;
    right: 5px;
    background: rgba(255,0,0,0.8);
    border: none;
    border-radius: 50%;
    color: white;
    width: 25px;
    height: 25px;
    font-size: 16px;
    cursor: pointer;
    z-index: 2;
  }
</style>
```

```
</head>
<body>
  <!-- Pantalla de bienvenida -->
  <div id="welcomeOverlay">
    <h1>Bienvenido a la Galería Compartida</h1>
    <p>Para comenzar, otorga permiso a tu cuenta de Google para acceder a
tu galería. Toca el botón para iniciar la autenticación.</p>
    <button class="welcomeButton" id="btnIniciar">Toca para
comenzar</button>
  </div>

  <!-- Interfaz de administración (oculta hasta autenticarse) -->
  <div class="fixed-header" id="header">
    <button class="button" id="authButton">Reautenticar</button>
    <button class="button" id="selectPhotosButton" disabled>Mostrar
Fotos</button>
    <button class="button" id="saveSelectionButton" disabled>Guardar
selección y abrir marco</button>
    <button class="button" id="manageReproductionButton"
disabled>Gestionar reproducción</button>
    <div id="progressBar"></div>
  </div>

  <div id="photosContainer"></div>
  <div id="message"></div>

  <!-- Overlay de elección de modo de uso -->
  <div id="usageScreen">
    <h2>¿Cómo se usará este dispositivo?</h2>
    <button class="usageButton" id="btnAdministrar">Administrar fotos y
videos</button>
    <button class="usageButton secondary" id="btnMarco">Usar como marco de
fotos</button>
  </div>

  <!-- Overlay para gestionar reproducción: muestra fotos seleccionadas
con opción de eliminación -->
  <div id="manageOverlay">
    <h2>Gestionar reproducción</h2>
    <div class="grid" id="manageGrid"></div>
```

```
    <button class="closeButton" id="closeManageOverlay">Cerrar</button>
  </div>

  <script>
    // Función para mostrar mensajes
    function showMessage(text) {
      console.log("Mensaje:", text);
      const messageDiv = document.getElementById('message');
      messageDiv.textContent = text;
      setTimeout(() => messageDiv.textContent = '', 5000);
    }

    // Configuración de la app
    const config = {
      CLIENT_ID:
'398620721557-cc1g475hejmglorrf7v2aa61c4btod9g.apps.googleusercontent.com'
,
      API_KEY: 'AIzaSyCcgAXaMxPnwPjqbWYpWFvDtfohrE6fRHI',
      SCOPES: 'https://www.googleapis.com/auth/photoslibrary.readonly'
    };

    let tokenClient;
    let accessToken = null;
    let selectedPhotos = new Set();
    let allFetchedPhotos = [];

    // Función para obtener la selección ya guardada (array de objetos
{id, url})
    function getStoredSelection() {
      const stored = localStorage.getItem('selectedPhotos');
      return stored ? JSON.parse(stored) : [];
    }

    // Inicializa GAPI
    async function gapiLoaded() {
      try {
        await new Promise((resolve, reject) => {
          gapi.load('client:auth2', {
            callback: resolve,
            ontimeout: () => reject(new Error('Timeout cargando GAPI')),
```

```
            timeout: 5000
          });
        });
        await gapi.client.init({
          apiKey: config.API_KEY,
          discoveryDocs:
['https://photoslibrary.googleapis.com/$discovery/rest?version=v1']
        });
        console.log('GAPI inicializado');
      } catch (error) {
        console.error('Error inicializando GAPI:', error);
        showMessage(`Error técnico: ${error.message}`);
      }
    }

    // Inicializa el cliente OAuth2
    function gisLoaded() {
      tokenClient = google.accounts.oauth2.initTokenClient({
        client_id: config.CLIENT_ID,
        scope: config.SCOPES,
        callback: (tokenResponse) => {
          if (tokenResponse?.access_token) {
            accessToken = tokenResponse.access_token;
            document.getElementById('selectPhotosButton').disabled =
false;
            document.getElementById('manageReproductionButton').disabled =
false;
            showMessage('¡Autenticación exitosa!');
            // Oculta el overlay de bienvenida, muestra el header y la
pantalla de uso
            document.getElementById('welcomeOverlay').style.display =
'none';
            document.getElementById('header').style.display = 'flex';
            showUsageScreen();
          }
        },
        error_callback: (error) => {
          let message = 'Error de autenticación';
          if (error.type === 'popup_closed') {
```

```
            message = 'La ventana emergente se cerró antes de completar la
autenticación. Por favor, inténtalo de nuevo.';
          } else if (error.type === 'user_logged_out') {
            message = 'Debe iniciar sesión en Google primero.';
          }
          showMessage(message);
          console.error("Error de autenticación:", error);
        }
      });
    }

    // Función para iniciar autenticación
    function iniciarAutenticacion() {
      if (!tokenClient) {
        showMessage('Sistema no inicializado. Recarga la página.');
        return;
      }
      tokenClient.requestAccessToken({ prompt: 'select_account' });
    }

    // Muestra el overlay de elección de modo de uso
    function showUsageScreen() {
      console.log("Mostrando pantalla de uso");
      const usageScreen = document.getElementById('usageScreen');
      usageScreen.style.display = 'flex';
      document.getElementById('btnAdministrar').onclick = () => {
        console.log("Modo administrar seleccionado");
        usageScreen.style.display = 'none';
      };
      document.getElementById('btnMarco').onclick = () => {
        console.log("Modo marco seleccionado");
        usageScreen.style.display = 'none';
        // Redirige a clienteb.html en la misma pestaña
        window.location.href = 'clienteb.html';
      };
    }

    // Función para obtener fotos desde Google Photos
    async function fetchPhotos() {
      if (!accessToken) {
```

```
        showMessage('No hay token de acceso. Autentícate primero.');
        return;
      }
      const progressBar = document.getElementById('progressBar');
      const container = document.getElementById('photosContainer');
      const button = document.getElementById('selectPhotosButton');
      try {
        button.disabled = true;
        container.innerHTML = '';
        progressBar.style.width = '0%';
        let photos = [];
        let nextPageToken = null;
        let pageCount = 0;
        do {
          const params = new URLSearchParams({
            pageSize: 100,
            pageToken: nextPageToken || ''
          });
          const response = await
fetch(`https://photoslibrary.googleapis.com/v1/mediaItems?${params.toStrin
g()}`, {
            headers: { Authorization: `Bearer ${accessToken}` }
          });
          if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
          }
          const data = await response.json();
          photos = photos.concat(data.mediaItems || []);
          nextPageToken = data.nextPageToken;
          pageCount++;
          progressBar.style.width = `${Math.min(pageCount * 10, 95)}%`;
        } while (nextPageToken && pageCount < 10);
        progressBar.style.width = '100%';
        allFetchedPhotos = photos;
        // Filtra las fotos que ya están en la reproducción (almacenadas
en localStorage)
        const stored = getStoredSelection();
        const selectedIds = new Set(stored.map(item => item.id));
        const available = photos.filter(photo =>
!selectedIds.has(photo.id));
```

```
      displayPhotos(available);
    } catch (error) {
      console.error('Error al cargar fotos:', error);
      showMessage('Error al cargar fotos. Intente nuevamente.');
    } finally {
      button.disabled = false;
      setTimeout(() => progressBar.style.width = '0%', 1000);
    }
  }

  // Muestra las fotos en pantalla
  function displayPhotos(photos) {
    const container = document.getElementById('photosContainer');
    container.innerHTML = '';
    if (!photos.length) {
      container.textContent = 'No se encontraron fotos disponibles.';
      return;
    }
    photos.forEach(photo => {
      const img = document.createElement('img');
      if (photo.baseUrl) {
        img.src = photo.baseUrl + '=w150-h150';
      } else {
        img.alt = photo.filename || 'Foto sin URL';
        img.style.background = '#ccc';
        img.style.padding = '10px';
        img.textContent = photo.filename || 'Sin nombre';
      }
      img.title = photo.filename || '';
      // Al hacer clic se alterna la selección
      img.onclick = () => toggleSelection(img, photo.id);
      container.appendChild(img);
    });
  }

  // Alterna la selección de una foto
  function toggleSelection(img, photoId) {
    const isSelected = img.hasAttribute('selected');
    if (isSelected) {
      img.removeAttribute('selected');
```

```
        selectedPhotos.delete(photoId);
      } else {
        img.setAttribute('selected', 'true');
        selectedPhotos.add(photoId);
      }
      document.getElementById('saveSelectionButton').disabled =
selectedPhotos.size === 0;
    }

    // Guarda la selección en localStorage como array de objetos { id, url
}
    function saveSelectedPhotos() {
      if (!allFetchedPhotos.length) {
        showMessage('No hay fotos cargadas para guardar.');
        return;
      }
      const selected = allFetchedPhotos.filter(photo =>
selectedPhotos.has(photo.id))
                                        .map(photo => ({
                                          id: photo.id,
                                          url: photo.baseUrl +
'=w1920-h1080'
                                        }));
      if (selected.length === 0) {
        showMessage('No se seleccionó ninguna foto.');
        return;
      }
      localStorage.setItem('selectedPhotos', JSON.stringify(selected));
      showMessage('Selección guardada. Puedes reproducirla en otro
dispositivo.');
      // Actualiza la lista de miniaturas, quitando las seleccionadas.
      fetchPhotos();
    }

    // Muestra el overlay para gestionar la reproducción (eliminar fotos
de la lista actual)
    function showManageOverlay() {
      const manageOverlay = document.getElementById('manageOverlay');
      const grid = document.getElementById('manageGrid');
      grid.innerHTML = '';
```

```
      const stored = getStoredSelection();
      if (stored.length === 0) {
        grid.textContent = 'No hay fotos cargadas para reproducción.';
      } else {
        stored.forEach(item => {
          const div = document.createElement('div');
          div.style.position = 'relative';
          const img = new Image();
          img.src = item.url;
          img.style.width = '100%';
          img.style.height = '150px';
          img.style.objectFit = 'cover';
          img.style.borderRadius = '8px';
          // Botón para eliminar
          const btnDelete = document.createElement('button');
          btnDelete.textContent = '×';
          btnDelete.classList.add('deleteIcon');
          btnDelete.onclick = () => {
            removeFromReproduction(item.id);
          };
          div.appendChild(img);
          div.appendChild(btnDelete);
          grid.appendChild(div);
        });
      }
      manageOverlay.style.display = 'flex';
    }

    // Elimina un medio de la lista de reproducción (actualiza
localStorage)
    function removeFromReproduction(id) {
      let stored = getStoredSelection();
      stored = stored.filter(item => item.id !== id);
      localStorage.setItem('selectedPhotos', JSON.stringify(stored));
      showMessage('Foto eliminada de la reproducción.');
      // Actualiza el overlay de gestión
      showManageOverlay();
      // Vuelve a recargar las miniaturas disponibles
      fetchPhotos();
    }
```

```
    // Inicialización de la gestión de reproducción: configura botón de
gestión
    function initManageButton() {
      const btnManage =
document.getElementById('manageReproductionButton');
      btnManage.onclick = () => {
        showManageOverlay();
      };
    }

    // Inicialización al cargar la página
    window.onload = async () => {
      document.getElementById('header').style.display = 'none';
      try {
        await gapiLoaded();
        gisLoaded();
      } catch (error) {
        console.error('Error de inicialización:', error);
        showMessage('Error crítico. Recarga la aplicación.');
      }
      // Configura botones de la interfaz
      document.getElementById('authButton').onclick =
iniciarAutenticacion;
      document.getElementById('selectPhotosButton').onclick = fetchPhotos;
      document.getElementById('saveSelectionButton').onclick =
saveSelectedPhotos;
      initManageButton();
      // Configura el botón de bienvenida
      document.getElementById('btnIniciar').onclick = () => {
        iniciarAutenticacion();
      };
      // Configura el botón de cerrar en el overlay de gestión
      document.getElementById('closeManageOverlay').onclick = () => {
        document.getElementById('manageOverlay').style.display = 'none';
      };
    };
  </script>
  <!-- Overlay para gestión de reproducción -->
  <div id="manageOverlay">
```

```
    <h2>Gestionar reproducción</h2>
    <div class="grid" id="manageGrid"></div>
    <button class="closeButton" id="closeManageOverlay">Cerrar</button>
  </div>
</body>
</html>
```

server

```
const express = require('express');
const path = require('path');
const helmet = require('helmet');
const rateLimit = require('express-rate-limit');
const cors = require('cors');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(
  helmet({
    contentSecurityPolicy: {
      directives: {
        defaultSrc: ["'self'"],
        scriptSrc: [
          "'self'",
          'https://apis.google.com',
          'https://accounts.google.com',
          "'unsafe-inline'"
        ],
        // Permite cargar imágenes desde 'self', data: y cualquier dominio
HTTPS de Google
        imgSrc: ["'self'", 'data:', 'https://*.googleusercontent.com'],
        frameSrc: [
          "'self'",
          'https://accounts.google.com',
          'https://content.googleapis.com',
          'https://content-photoslibrary.googleapis.com'
```

```
        ],
        connectSrc: [
          "'self'",
          'https://*.googleapis.com',
          'https://*.google.com'
        ]
      }
    },
    // Permite que las ventanas emergentes se cierren sin interferencia
    crossOriginOpenerPolicy: { policy: "same-origin-allow-popups" }
  })
);

app.use(cors());
app.use(express.json());

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000,
  max: 100
});
app.use(limiter);

app.use(express.static(path.join(__dirname), {
  setHeaders: (res) => {
    res.set('Cache-Control', 'public, max-age=86400');
  }
}));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});

app.listen(PORT, () =>
  console.log(`Servidor corriendo en http://localhost:${PORT}`)
);
```

NO REPRODUCE VIDEOS