

Migrador de Carpetas

Tenemos una carpeta en Google Drive que contiene archivos y carpetas y más archivos y más carpetas.... Y quiero hacer una copia en otro drive de otra cuenta que no pertenece a mi dominio.

Hasta ahora o buscas soluciones de terceros, o haces un zip, lo descargas, lo subes al destino y lo descomprimes allí, una foto finish.

Pero y ¿si compartimos con el correo externo la carpeta que queremos salvaguardar? Esto nos daría acceso directo a la carpeta del drive origen desde el drive destino, ... hasta que dejen de compartirla con nuestra cuenta.

Realmente no tenemos la carpeta en nuestro drive de destino y las alternativas para hacerse una copia son las que hemos comentado antes, si la carpeta no contiene mucha estructura es accesible hacerlo a mano pero ¿y si no es el caso?

Con esta solución podemos crear una carpeta copia de la original compartida en una ubicación de su propio drive y elaborar un informe del proceso de copia, también haremos que si el script se para o se interrumpe, podremos pedirle que continúe por donde se quedó.

Script de Migrador de carpetas en el drive de destino

Código.gs

```
const NOMBRE_CARPETA_INFORMES = "Informes de copia Drive";

function doGet() {
  return HtmlService.createHtmlOutputFromFile("FormularioDestino")
    .setTitle("Copia desde carpeta compartida");
}

function iniciarCopia(datos) {
  try {
    const idOrigen = datos.idCarpeta.trim();
    let nombreDestino = datos.nombreDestino.trim() || "Copia desde compartido";
    let modo = datos.modos?.trim() || "auto";
    const carpetaOrigen = DriveApp.getFolderById(idOrigen);
    const nombreRaizOrigen = carpetaOrigen.getName();

    let carpetaDestino;
    let mensajeEstado = "";
    const carpetas = DriveApp.getFoldersByName(nombreDestino);
```

```

let registro;

if (carpetas.hasNext()) {
  carpetaDestino = carpetas.next();

  if (modo === "auto") {
    if (carpetaDestino.GetFiles().hasNext() ||
carpetaDestino.getFolders().hasNext()) {
      const registroExistente = cargarRegistro(nombreDestino);
      if (Object.keys(registroExistente).length === 0) {
        modo = "continuar";
        registro = generarRegistroDesdeDestino(carpetaOrigen,
carpetaDestino);
        mensajeEstado = `⚠ Carpeta "${nombreDestino}" tenía contenido
pero sin registro previo. Se ha generado uno desde el contenido actual.`;
      } else {
        modo = "continuar";
      }
    } else {
      modo = "nuevo";
    }
  }

  if (modo === "nuevo" && (carpetaDestino.GetFiles().hasNext() ||
carpetaDestino.getFolders().hasNext())) {
    modo = "continuar";
    mensajeEstado = `⚠ Carpeta "${nombreDestino}" ya tenía contenido.
Cambiado a modo CONTINUAR.`;
  } else if (!mensajeEstado) {
    mensajeEstado = `Usando carpeta existente: "${nombreDestino}".`;
  }

} else {
  carpetaDestino = DriveApp.createFolder(nombreDestino);
  mensajeEstado = `Carpeta creada: "${nombreDestino}".`;
}

```

```

    registro = modo === "continuar" && typeof registro === "undefined" ?
cargarRegistro(nombreDestino) : (registro || {});

    const nuevos = copiarRecursoConProgreso(carpetaOrigen, carpetaDestino,
registro, "", nombreRaizOrigen);

    let mensajeFinal = nuevos === 0
    ? "✅ Todo ya estaba copiado. No se han hecho cambios."
    : `🟡 Se han copiado ${nuevos} elementos nuevos.`;

    const nombreArchivoRegistro = generarNombreRegistro(nombreDestino);
    registro.__fecha = Utilities.formatDate(new Date(),
Session.getScriptTimeZone(), "yyyy-MM-dd HH:mm:ss");
    const archivoRegistro = guardarRegistro(registro,
nombreArchivoRegistro);

    let advertencias = "";
    if (registro.__pesados?.length > 0) {
        advertencias += "⚠ Archivos grandes detectados:<br>" +
registro.__pesados.map(p => "- " + p).join("<br>") + "<br><br>";
    }
    if (registro.__errores?.length > 0) {
        advertencias += "❌ Errores durante la copia:<br>" +
registro.__errores.map(e => "- " + e).join("<br>") + "<br><br>";
    }

    let mensajeHTML = "";
    mensajeHTML += `<div>📁 ${nombreDestino}</div>`;
    mensajeHTML += `<div>${mensajeEstado}<br>${mensajeFinal}</div>`;

    if (advertencias.trim()) {
        mensajeHTML += `<div style="margin-top: 10px">${advertencias}</div>`;
    }

    return {
        mensaje: mensajeHTML,
        idDestino: carpetaDestino.getId(),
        nombreDestino: nombreDestino,
        urlRegistro: archivoRegistro.getUrl()
    }

```

```

    };

    } catch (e) {
        Logger.log("Error general en iniciarCopia: " + e.message);
        return { mensaje: "✗ Ocurrió un error inesperado. Revisa los registros." };
    }
}

function copiarRecursoConProgreso(origen, destino, registro, rutaBase,
raiz) {
    const nombreActual = origen.getName();
    const rutaActual = rutaBase ? `${rutaBase}/${nombreActual}` :
`${nombreActual}`;

    if (!registro[rutaActual]) registro[rutaActual] = { archivos: {},
carpetas: {} };
    if (!registro.__errores) registro.__errores = [];
    if (!registro.__pesados) registro.__pesados = [];
    if (!registro.__raiz) registro.__raiz = raiz;

    let nuevosCopiados = 0;

    const archivosDestino = destino.GetFiles();
    const mapaArchivosDestino = {};
    while (archivosDestino.hasNext()) {
        const archivo = archivosDestino.next();
        mapaArchivosDestino[archivo.getName()] = archivo;
    }

    const archivosOrigen = origen.GetFiles();
    while (archivosOrigen.hasNext()) {
        const archivo = archivosOrigen.next();
        const nombre = archivo.getName();
        const size = archivo.getSize();
        const modificado = archivo.getLastUpdated().getTime();

        const yaRegistrado = registro[rutaActual].archivos[nombre];

```

```

const mismoContenido = yaRegistrado &&
                        yaRegistrado.size === size &&
                        yaRegistrado.modificado === modificado;

const existeEnDestino = !!mapaArchivosDestino[nombre];

// 🔄 Si está registrado pero ha sido eliminado en destino, debe
recopiarse
const necesitaCopiar = !mismoContenido || !existeEnDestino;

if (necesitaCopiar) {
  if (size > 100 * 1024 * 1024) {
    registro.__pesados.push(`${rutaActual}/${nombre} (${Math.round(size
/ 1024 / 1024)} MB)`);
  }
  try {
    if (existeEnDestino) {
      mapaArchivosDestino[nombre].setTrashed(true);
    }
    const copia = archivo.makeCopy(nombre, destino);
    registro[rutaActual].archivos[nombre] = {
      id: copia.getId(),
      size: size,
      modificado: modificado
    };
    nuevosCopiados++;
  } catch (e) {
    const msg = `Error al copiar archivo "${nombre}" en ${rutaActual}:
${e.message}`;
    Logger.log(msg);
    registro.__errores.push(msg);
  }
}

const subcarpetasDestino = destino.getFolders();
const mapaCarpetasDestino = {};
while (subcarpetasDestino.hasNext()) {
  const c = subcarpetasDestino.next();

```

```

    mapaCarpetasDestino[c.getName()] = c.getId();
}

const subcarpetasOrigen = origen.getFolders();
while (subcarpetasOrigen.hasNext()) {
    const sub = subcarpetasOrigen.next();
    const nombreSub = sub.getName();

    const idDestino = registro[rutaActual].carpetas[nombreSub];
    const existeSubEnDestino = idDestino && mapaCarpetasDestino[nombreSub];

    if (!idDestino || !existeSubEnDestino) { // ↻ incluir creación si ha
    sido eliminada
        try {
            const nuevaSub = destino.createFolder(nombreSub);
            registro[rutaActual].carpetas[nombreSub] = nuevaSub.getId();
            nuevosCopiados++;
        } catch (e) {
            const msg = `Error al crear carpeta "${nombreSub}" en ${rutaActual}:
            ${e.message}`;
            Logger.log(msg);
            registro.__errores.push(msg);
            continue;
        }
    }

    try {
        const subDestino =
DriveApp.getFolderById(registro[rutaActual].carpetas[nombreSub]);
        nuevosCopiados += copiarRecursivoConProgreso(sub, subDestino,
registro, rutaActual, raiz);
    } catch (e) {
        const msg = `Error al continuar con subcarpeta "${nombreSub}" en
${rutaActual}: ${e.message}`;
        Logger.log(msg);
        registro.__errores.push(msg);
    }
}
}

```

```

return nuevosCopiados;
}

function verificarIntegridad(datos) {
  try {
    const idOrigen = datos.idCarpeta.trim();
    const nombreDestino = datos.nombreDestino.trim();
    const carpetaOrigen = DriveApp.getFolderById(idOrigen);

    const carpetas = DriveApp.getFoldersByName(nombreDestino);
    if (!carpetas.hasNext()) {
      return `❌ No se encontró la carpeta de destino "${nombreDestino}".`;
    }

    const carpetaDestino = carpetas.next();
    const registro = cargarRegistro(nombreDestino);
    const nombreRaizOrigen = carpetaOrigen.getName();
    const resultado = compararEstructuras(carpetaOrigen, carpetaDestino, "",
registro, nombreRaizOrigen);
    //const resultado = compararEstructuras(carpetaOrigen, carpetaDestino,
"", registro);

    if (resultado.trim() === "") {
      return "✅ Todo coincide: no se encontraron diferencias entre origen y
destino.";
    } else {
      return `
        <strong>⚠ Verificación finalizada con
incidencias:</strong><br><br>${resultado}
        <br><br>🔄 Puedes volver a lanzar la copia para intentar completar o
actualizar los elementos.
        <br><br>✂ Si deseas eliminar del destino los elementos que ya no
están en el origen, usa el botón de <strong>Limpiar</strong>.`;
    }
  } catch (e) {
    return `❌ Error al verificar integridad: " + e.message;
  }
}

```

```

}

function compararEstructuras(origen, destino, rutaBase, registro, raiz) {
  const rutaActual = rutaBase ? `${rutaBase}/${origen.getName()}` :
`${raiz}`;
  let resultado = "";

  const registroRuta = registro[rutaActual]?.archivos || {};
  const carpetasRegistradas = registro[rutaActual]?.carpetas || {};

  const archivosOrigen = origen.GetFiles();
  const mapaOrigen = {};
  const detallesOrigen = {};
  while (archivosOrigen.hasNext()) {
    const f = archivosOrigen.next();
    const nombre = f.getName();
    mapaOrigen[nombre] = f.getSize();
    detallesOrigen[nombre] = f.getLastUpdated().getTime();
  }

  const archivosDestino = destino.GetFiles();
  const nombresDestino = new Set();
  while (archivosDestino.hasNext()) {
    nombresDestino.add(archivosDestino.next().getName());
  }

  for (const nombre in mapaOrigen) {
    const size = mapaOrigen[nombre];
    const modificado = detallesOrigen[nombre];

    if (!(nombre in registroRuta)) {
      resultado += `❌ Archivo no registrado aún:
${rutaActual}/${nombre}<br>`;
    } else {
      const reg = registroRuta[nombre];
      try {
        const archivoDestino = DriveApp.getFileById(reg.id);
        if (archivoDestino.isTrashed()) {

```



```

        resultado += `❌ Archivo eliminado (en papelera):
${rutaActual}/${nombre}<br>`;
    }
    } catch (e) {
        resultado += `❌ Archivo faltante en destino:
${rutaActual}/${nombre}<br>`;
        continue;
    }

    if (reg.size !== size) {
        resultado += `⚠️ Tamaño cambiado: ${rutaActual}/${nombre} (antes:
${reg.size}, ahora: ${size})<br>`;
    } else if (reg.modificado !== modificado) {
        resultado += `⚠️ Modificado desde la copia: ${rutaActual}/${nombre}
(original: ${new Date(reg.modificado).toLocaleString()}, ahora: ${new
Date(modificado).toLocaleString()})<br>`;
    }
    }
}

for (const nombre of nombresDestino) {
    if (!(nombre in mapaOrigen)) {
        resultado += `⚠️ Archivo en destino que ya no está en origen:
${rutaActual}/${nombre}<br>`;
    }
}

// Verificar subcarpetas registradas eliminadas
for (const nombre in carpetasRegistradas) {
    const id = carpetasRegistradas[nombre];
    try {
        const carpeta = DriveApp.getFolderById(id);
        if (carpeta.isTrashed()) {
            resultado += `❌ Subcarpeta eliminada (en papelera):
${rutaActual}/${nombre}<br>`;
        }
    } catch (e) {
        resultado += `❌ Subcarpeta faltante en destino:
${rutaActual}/${nombre}<br>`;
    }
}

```

```

    }
}

const subcarpetasOrigen = origen.getFolders();
const mapaSubOrigen = {};
while (subcarpetasOrigen.hasNext()) {
    const sub = subcarpetasOrigen.next();
    mapaSubOrigen[sub.getName()] = sub;
}

const subcarpetasDestino = destino.getFolders();
const mapaSubDestino = {};
while (subcarpetasDestino.hasNext()) {
    const sub = subcarpetasDestino.next();
    mapaSubDestino[sub.getName()] = sub;
}

for (const nombre in mapaSubOrigen) {
    if (!(nombre in mapaSubDestino)) {
        resultado += `❌ Falta subcarpeta en destino:
${rutaActual}/${nombre}<br>`;
    } else {
        resultado += compararEstructuras(mapaSubOrigen[nombre],
mapaSubDestino[nombre], rutaActual, registro, raiz);
    }
}

for (const nombre in mapaSubDestino) {
    if (!(nombre in mapaSubOrigen)) {
        resultado += `⚠️ Carpeta en destino que ya no está en origen:
${rutaActual}/${nombre}<br>`;
    }
}

return resultado;
}

function generarNombreRegistro(nombreDestino) {

```

```

const ahora = new Date();
const pad = n => n.toString().padStart(2, '0');
const marca = `${ahora.getFullYear()}-${pad(ahora.getMonth() +
1)}-${pad(ahora.getDate())}-${pad(ahora.getHours())}-${pad(ahora.getMinutes
())}`;
const nombreLimpiado = nombreDestino.replace(/^[^w\d-]/g, "_");
return `progreso_copia_drive_${nombreLimpiado}_${marca}.json`;
}

function guardarRegistro(registro, nombreArchivo) {
  try {
    const json = JSON.stringify(registro, null, 2);
    const carpetaInformes = obtenerOCrearCarpeta(NOMBRE_CARPETA_INFORMES);
    const archivo = carpetaInformes.createFile(nombreArchivo, json,
MimeType.PLAIN_TEXT);
    return archivo;
  } catch (e) {
    Logger.log("Error al guardar el registro: " + e.message);
    return null;
  }
}

function cargarRegistro(nombreDestino) {
  try {
    const carpetaInformes = obtenerOCrearCarpeta(NOMBRE_CARPETA_INFORMES);
    const archivos = carpetaInformes.GetFiles();
    let ultimoArchivo = null;
    let ultimaFecha = 0;

    const nombreLimpiado = nombreDestino.replace(/^[^w\d-]/g, "_");
    const prefijo = `progreso_copia_drive_${nombreLimpiado}`;

    while (archivos.hasNext()) {
      const archivo = archivos.next();
      const nombre = archivo.getName();

      if (!nombre.startsWith(prefijo)) continue;

      const fecha = archivo.getLastUpdated().getTime();

```

```

        if (fecha > ultimaFecha) {
            ultimaFecha = fecha;
            ultimoArchivo = archivo;
        }
    }

    if (!ultimoArchivo) return {};
    const contenido = ultimoArchivo.getBlob().getDataAsString();
    return JSON.parse(contenido);
} catch (e) {
    Logger.log("Error al cargar el registro: " + e.message);
    return {};
}
}

function obtenerOCrearCarpeta(nombre) {
    const carpetas = DriveApp.getFoldersByName(nombre);
    return carpetas.hasNext() ? carpetas.next() :
    DriveApp.createFolder(nombre);
}

function limpiarDestino(datos) {
    try {
        const idOrigen = datos.idCarpeta.trim();
        const nombreDestino = datos.nombreDestino.trim();
        const carpetaOrigen = DriveApp.getFolderById(idOrigen);

        const carpetas = DriveApp.getFoldersByName(nombreDestino);
        if (!carpetas.hasNext()) {
            return `❌ No se encontró la carpeta de destino "${nombreDestino}";`
        }

        const carpetaDestino = carpetas.next();
        const registroEliminados = [];

        eliminarSobrantes(carpetaOrigen, carpetaDestino, "",
registroEliminados);

        if (registroEliminados.length === 0) {

```

```

    return "✅ No había elementos sobrantes. Nada se ha eliminado.";
  } else {
    const resumen = registroEliminados.map(e => "- " + e).join("<br>");
    return `🧹 Se han eliminado ${registroEliminados.length} elementos
sobrantes:<br><br>${resumen}`;
  }

} catch (e) {
  return "❌ Error durante la limpieza: " + e.message;
}
}

```

```

function eliminarSobrantes(origen, destino, rutaBase, eliminados) {
  const rutaActual = rutaBase + "/" + origen.getName();

  // Archivos
  const archivosOrigen = origen.GetFiles();
  const setOrigen = new Set();
  while (archivosOrigen.hasNext()) {
    setOrigen.add(archivosOrigen.next().getName());
  }

  const archivosDestino = destino.GetFiles();
  while (archivosDestino.hasNext()) {
    const f = archivosDestino.next();
    const nombre = f.getName();
    if (!setOrigen.has(nombre)) {
      try {
        f.setTrashed(true);
        eliminados.push(`Archivo eliminado del destino:
${rutaActual}/${nombre}`);
      } catch (e) {
        eliminados.push(`❌ Error al eliminar archivo:
${rutaActual}/${nombre} → ${e.message}`);
      }
    }
  }
}

```

```

// Subcarpetas
const subcarpetasOrigen = origen.getFolders();
const setSubOrigen = new Set();
while (subcarpetasOrigen.hasNext()) {
    setSubOrigen.add(subcarpetasOrigen.next().getName());
}

const subcarpetasDestino = destino.getFolders();
while (subcarpetasDestino.hasNext()) {
    const sub = subcarpetasDestino.next();
    const nombre = sub.getName();
    if (!setSubOrigen.has(nombre)) {
        try {
            sub.setTrashed(true);
            eliminados.push(`Carpeta eliminada del destino:
${rutaActual}/${nombre}`);
        } catch (e) {
            eliminados.push(`❌ Error al eliminar carpeta:
${rutaActual}/${nombre} → ${e.message}`);
        }
    } else {
        const subOrigen = origen.getFoldersByName(nombre);
        if (subOrigen.hasNext()) {
            eliminarSobrantes(subOrigen.next(), sub, rutaActual, eliminados);
        }
    }
}

function generarRegistroDesdeDestino(origen, destino) {
    const registro = { __errores: [], __pesados: [], __raiz: origen.getName() };
    construirRegistroDesdeDestino(origen, destino, registro, "");
    return registro;
}

function construirRegistroDesdeDestino(origen, destino, registro, rutaBase) {
    const nombreActual = origen.getName();

```

```
const rutaActual = rutaBase ? `${rutaBase}/${nombreActual}` :
`${nombreActual}`;
registro[rutaActual] = { archivos: {}, carpetas: {} };

const archivosOrigen = origen.GetFiles();
const mapaOrigen = {};
while (archivosOrigen.hasNext()) {
    const a = archivosOrigen.next();
    mapaOrigen[a.getName()] = a.getSize();
}

const archivosDestino = destino.GetFiles();
while (archivosDestino.hasNext()) {
    const f = archivosDestino.next();
    const nombre = f.getName();
    const size = f.getSize();
    if (nombre in mapaOrigen && mapaOrigen[nombre] === size) {
        registro[rutaActual].archivos[nombre] = {
            id: f.getId(),
            size: size,
            modificado: null // Se ignorará en la siguiente comprobación
        };
    }
}

const subcarpetasOrigen = origen.getFolders();
const mapaSubOrigen = {};
while (subcarpetasOrigen.hasNext()) {
    const sub = subcarpetasOrigen.next();
    mapaSubOrigen[sub.getName()] = sub;
}

const subcarpetasDestino = destino.getFolders();
while (subcarpetasDestino.hasNext()) {
    const sub = subcarpetasDestino.next();
    const nombre = sub.getName();
    if (nombre in mapaSubOrigen) {
        registro[rutaActual].carpetas[nombre] = sub.getId();
    }
}
```

```

        construirRegistroDesdeDestino(mapaSubOrigen[nombre], sub, registro,
rutaActual);
    }
}
}

```

FormularioDestino.html

```

<!DOCTYPE html>
<html>
<head>
<base target="_top">
<style>
    body {
        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
        margin: 0;
        padding: 0;
        height: 100vh;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .contenedor {
        background-color: white;
        padding: 30px;
        border-radius: 8px;
        box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
        width: 100%;
        max-width: 460px;
    }

    h2 {
        text-align: center;
        margin-bottom: 20px;
    }

    label {

```



```
    font-weight: bold;
    margin-top: 15px;
    display: block;
}

input, button {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    background-color: #4285F4;
    color: white;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #3367D6;
}

.spinner {
    display: none;
    margin: 20px auto;
    border: 5px solid #f3f3f3;
    border-top: 5px solid #4285F4;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    animation: girar 1s linear infinite;
}

@keyframes girar {
    0% { transform: rotate(0deg); }
```

```
100% { transform: rotate(360deg); }  
}
```

```
#resultado {  
  margin-top: 20px;  
  font-size: 0.95em;  
  text-align: left;  
}
```

```
.nota {  
  font-size: 0.9em;  
  color: #666;  
  text-align: center;  
  margin-top: -10px;  
  margin-bottom: 15px;  
}
```

```
.mensaje-ok {  
  background-color: #e8f5e9;  
  color: #1b5e20;  
  padding: 10px;  
  border-left: 4px solid #2e7d32;  
  border-radius: 4px;  
}
```

```
.mensaje-error {  
  background-color: #ffebee;  
  color: #b71c1c;  
  padding: 10px;  
  border-left: 4px solid #c62828;  
  border-radius: 4px;  
}
```

```
.mensaje-aviso {  
  background-color: #fff8e1;  
  color: #e65100;  
  padding: 10px;  
  border-left: 4px solid #f9a825;  
  border-radius: 4px;  
}
```

```

    }
</style>
</head>
<body>
  <div class="contenedor">
    <h2>Copia desde carpeta compartida</h2>

    <label for="idCarpeta">ID de la carpeta compartida:</label>
    <input type="text" id="idCarpeta" placeholder="Ej: 1a2b3c4d5e...">

    <label for="nombreDestino">Nombre de la carpeta destino:</label>
    <input type="text" id="nombreDestino" placeholder="Ej: Copia IA
2ºESO...">

    <p class="nota">La copia se reanudará automáticamente si ya existe
contenido.<br>No se sobrescriben ni se duplican archivos.</p>

    <button onclick="enviar()">📁 Ejecutar copia</button>
    <button onclick="verificar()">🔍 Verificar integridad</button>
    <button onclick="limpiar()">🧹 Limpiar destino</button>

    <div class="spinner" id="spinner"></div>
    <div id="resultado"></div>
  </div>

  <script>
    function enviar() {
      const datos = {
        idCarpeta: document.getElementById("idCarpeta").value.trim(),
        nombreDestino:
document.getElementById("nombreDestino").value.trim(),
        modo: "auto"
      };

      if (!datos.idCarpeta) {
        document.getElementById("resultado").innerHTML = "<div
class='mensaje-error'>Por favor, introduce un ID de carpeta.</div>";
        return;
      }

```

```

        document.getElementById("spinner").style.display = "block";
        document.getElementById("resultado").innerHTML = "Procesando
copia...";

        google.script.run
            .withSuccessHandler(function (res) {
                document.getElementById("spinner").style.display = "none";

                document.getElementById("resultado").innerHTML = `
                    <div class="mensaje-ok">${res.mensaje}</div>
                    <div style="text-align: center; margin-top: 20px;">
                        <button
onclick="window.open('https://drive.google.com/drive/folders/${res.idDestino}', '_blank')" style="margin-top:10px">
                            Abrir carpeta en Drive
                        </button><br><br>
                        <button onclick="window.open('${res.urlRegistro}',
'_blank')" style="margin-top:5px">
                            Ver archivo de progreso
                        </button>
                    </div>
                `;
            })
            .iniciarCopia(datos);
    }

    function verificar() {
        const datos = {
            idCarpeta: document.getElementById("idCarpeta").value.trim(),
            nombreDestino:
document.getElementById("nombreDestino").value.trim()
        };

        if (!datos.idCarpeta || !datos.nombreDestino) {
            document.getElementById("resultado").innerHTML = "<div
class='mensaje-error'>Introduce el ID de carpeta y nombre destino para
verificar.</div>";
            return;
        }
    }

```

```

    }

    document.getElementById("spinner").style.display = "block";
    document.getElementById("resultado").textContent = "Verificando
integridad...";

    google.script.run
    .withSuccessHandler(function (res) {
        document.getElementById("spinner").style.display = "none";

        let claseResultado = res.includes("✅") ? "mensaje-ok" :
            res.includes("❌") ? "mensaje-error" :
                "mensaje-aviso";

        document.getElementById("resultado").innerHTML = `<div
class="${claseResultado}">${res}</div>`;
    })
    .verificarIntegridad(datos);
}

function limpiar() {
    const datos = {
        idCarpeta: document.getElementById("idCarpeta").value.trim(),
        nombreDestino:
document.getElementById("nombreDestino").value.trim()
    };

    if (!datos.idCarpeta || !datos.nombreDestino) {
        document.getElementById("resultado").innerHTML = "<div
class='mensaje-error'>Introduce el ID de carpeta y nombre destino para
limpiar.</div>";
        return;
    }

    if (!confirm("⚠ ¿Estás seguro de que deseas eliminar los elementos
sobrantes del destino?")) {
        return;
    }
}

```

```

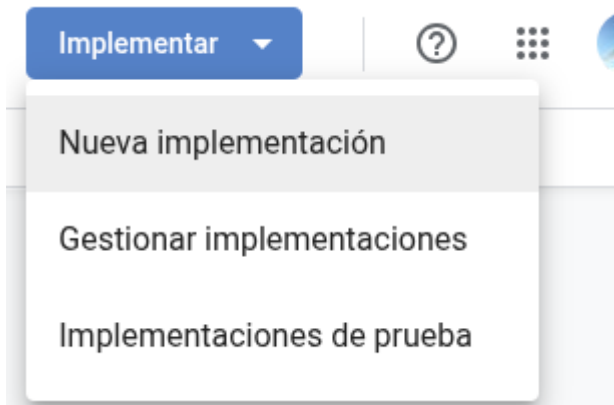
document.getElementById("spinner").style.display = "block";
document.getElementById("resultado").textContent = "Limpiando
destino...";

google.script.run
  .withSuccessHandler(function (res) {
    document.getElementById("spinner").style.display = "none";

    let claseResultado = res.includes("✅") ? "mensaje-ok" :
      res.includes("❌") ? "mensaje-error" :
        "mensaje-aviso";

    document.getElementById("resultado").innerHTML = `<div
class="${claseResultado}">${res}</div>`;
  })
  .limpiarDestino(datos);
}
</script>
</body>
</html>

```



Nueva implementación

Seleccionar tipo



Configuración



- Aplicación web
- Ejecutable de API
- Complemento
- Biblioteca



Selecciona un tipo de implementación

Cancelar

Implementar

Nueva implementación

Seleccionar tipo	Configuración
Aplicación web	<div>Descripción<div>Nueva descripción</div>Migrador carpetas web</div> <div>Aplicación web<div>Ejecutar como</div>Usuario que accede a la aplicación web</div> <div>Los usuarios tendrán que autorizar la ejecución de la aplicación web con los datos de su cuenta.</div> <div>Quién tiene acceso</div> Solo yo

También se puede usar como una biblioteca.[Más información](#)

Cancelar

Implementar

Nueva implementación

La implementación se ha actualizado correctamente.

Versión 3 del 3 may 2025, 13:16

ID de implementación

AKfycbxiKwMIYGjjaGd1HvtVT8JFO4O0RDkQ1jDgFezQLeZJauOntrg3NkrloGsg4zAraHEW

 Copiar

Aplicación web

URL

[https://script.google.com/a/macros/profe.tk/s/AKfycbxiKwMIYGjjaGd1HvtVT8JFO4O0RDkQ1jDgFezQLeZJauOntrg3NkrloGsg4zAraHEW...](https://script.google.com/a/macros/profe.tk/s/AKfycbxiKwMIYGjjaGd1HvtVT8JFO4O0RDkQ1jDgFezQLeZJauOntrg3NkrloGsg4zAraHEW/exec)

 Copiar

Hecho

El script que he puesto arriba es el siguiente

<https://script.google.com/a/macros/profe.tk/s/AKfycbxiKwMIYGjjaGd1HvtVT8JFO4O0RDkQ1jDgFezQLeZJauOntrg3NkrloGsg4zAraHEW/exec>


Nueva implementación

La aplicación web requiere que autorices el acceso a tus datos.

Autorizar acceso

Cancelar

Hecho

 Sign in with Google

Choose an account from profe.tk

to continue to [Migrador carpetas drive](#)



Antonio Varela García
profe@profe.tk



Use another account

 Sign in with Google

Migrador carpetas drive wants to access your Google Account

 profe@profe.tk

This will allow **Migrador carpetas drive** to:



See, edit, create, and delete all of your Google
Drive files



Make sure you trust Migrador carpetas drive



[Learn why you're not seeing links to Migrador
carpetas drive's Privacy Policy or Terms of Service](#)

Review Migrador carpetas drive's Privacy Policy and Terms of
Service to understand how Migrador carpetas drive will process and
protect your data.

To make changes at any time, go to your [Google Account](#).

Learn how Google helps you [share data safely](#).

Copia desde carpeta **compartida**

ID de la carpeta compartida:

Ej: 1a2b3c4d5e...

Nombre de la carpeta destino:


Ej: Copia IA 2ºESO...





La copia se reanuda automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.



 Ejecutar copia

 Verificar integridad

 Limpiar destino


→  https://drive.google.com/drive/u/1/folders/176sQ9DtavDStmW_T7o1KOH1PXZFdw4z

📁 Diseño 📁 Google 📁 Editores 📁 Utilidades 📁 Cursos Web 📁 Formación 📁 Ubuntu and Free S...  SPARKLINE - Ayuda...  Corubrics (es) - Fun...  IRIS Dashboard  Aula Virtual de F










 Drive 

+ Nuevo

- 🏠 Página principal
- 🔔 Actividad
- 👤 Espacios de trabajo
- 📁 Mi unidad
- 📁 Unidades compartidas
- ★ Destacados
- 👤 Compartido conmigo

Compartido conmigo > Mis clases 

Tipo Personas Modificado Fuente

Nombre ↑	Propietario	Última modificación ▼	Tamaño de a
 CyR	 antoniovarela	3 oct 2024	—
 DIG	 antoniovarela	12:26	—
 ej1	 antoniovarela	4 dic 2024	—
 myProject.zip 	 antoniovarela	4 dic 2024	922 bytes

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

- Ejecutar copia
- Verificar integridad
- Limpiar destino



Procesando copia...

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Carpeta creada: "Copia de Mis Clases".
 Se han copiado 13 elementos nuevos.

Abrir carpeta en Drive

Ver archivo de progreso

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Usando carpeta existente: "Copia de Mis Clases".
 Todo ya estaba copiado. No se han hecho cambios.

Abrir carpeta en Drive

Ver archivo de progreso

Ejemplo del registro Json generado como archivo de progreso

```
{
  "/Mis clases": {
    "archivos": {
      "myProject.zip": {
        "id": "1ec17kB_w4_3Plrbh71D6iEP14TiqN0Or",
        "size": 922,
        "modificado": 1733302024268
      }
    },
    "carpetas": {
      "DIG": "1JdgcepkJDPviRzh-kEWRYm_y4MSDtLsI",
```

```

    "ej1": "1DMS7v3BnpaN_PzgV84yng1zSdCiqH58R",
    "CyR": "15WGj18HuxsJm62km7u2u7qtzoCsKt91S"
  }
},
"_errores": [],
"_pesados": [],
"_raiz": "Mis clases",
"/Mis clases/DIG": {
  "archivos": {
    "4ESOC_Tuto2_-_Get_Started_2024_12_12_11_57_19.zip": {
      "id": "1M4zWJWgrpIdKW1NhAPy14jMFCQCNaPpn",
      "size": 5412822,
      "modificado": 1734004647129
    },
    "4esoC_Tutorial1_2024_12_11_13_19_03.zip": {
      "id": "1JDA7ETrL73htwkwR6MG5UJV9j9XuoQKA",
      "size": 5412414,
      "modificado": 1733923174209
    },
    "4ESOB_Tuto1_2024_12_11_08_49_27.zip": {
      "id": "1ifWUuqGV-MZgrHQbvsM9JMYJQhZ1AhvK",
      "size": 5212149,
      "modificado": 1733907006447
    }
  },
  "carpetas": {}
},
"/Mis clases/ej1": {
  "archivos": {
    "style.css": {
      "id": "1RpXGZ74uZ19SQPp18Ir_GFP6Khgc97Fp",
      "size": 70,
      "modificado": 1733302294186
    },
    "sketch.js": {
      "id": "1HZIy_CMU_VNixobT0519fz6TQkTehoHg",
      "size": 128,
      "modificado": 1733302294176
    },
    "index.html": {
      "id": "1LH5iptjTv42Rk9jojnrv4gkZavvPQexa",
      "size": 418,
      "modificado": 1733302294154
    }
  },
  "carpetas": {}
},
"/Mis clases/CyR": {
  "archivos": {
    "ESO3A- clones (2).sb3": {
      "id": "1hFQEJAq9ycXLyup4rIRJkMW24XbggV-z",
      "size": 64907,
      "modificado": 1733818673807
    },
    "ESO3A- clones (1).sb3": {
      "id": "18B26xIPF0lZNplu6sM4VHqlZksmbLJF5",
      "size": 36625,
      "modificado": 1733818338712
    }
  }
}

```

```
    },
    "ESO3A- clones.sb3": {
      "id": "1hMdublAEU6gQimzfl69KXsld89rvAuZp",
      "size": 64910,
      "modificado": 1733818107724
    }
  },
  "carpetas": {}
},
"__fecha": "2025-05-03 10:13:42"
}
```

Mi unidad > Copia de Mis Clases ▾

Tipo ▾

Personas ▾

Modificado ▾

Fuente ▾

Nombre ↑



CyR



DIG



ej1



myProject.zip

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino



Verificando integridad...

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

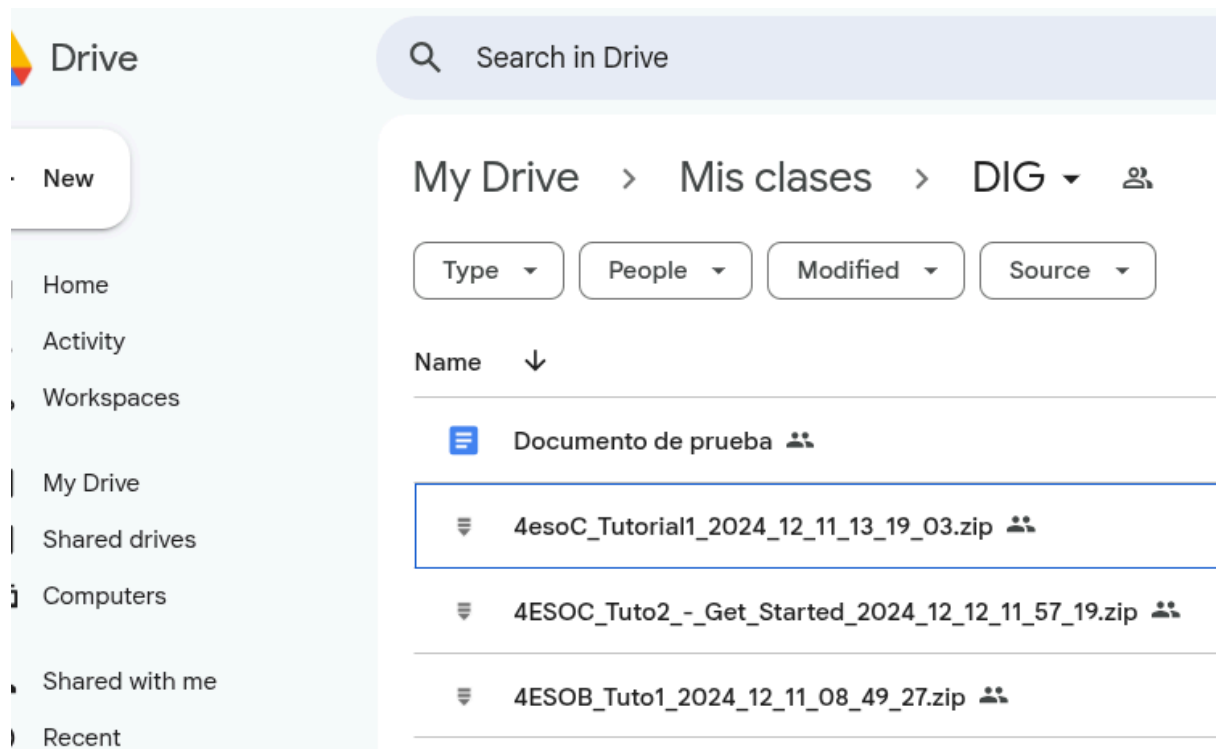
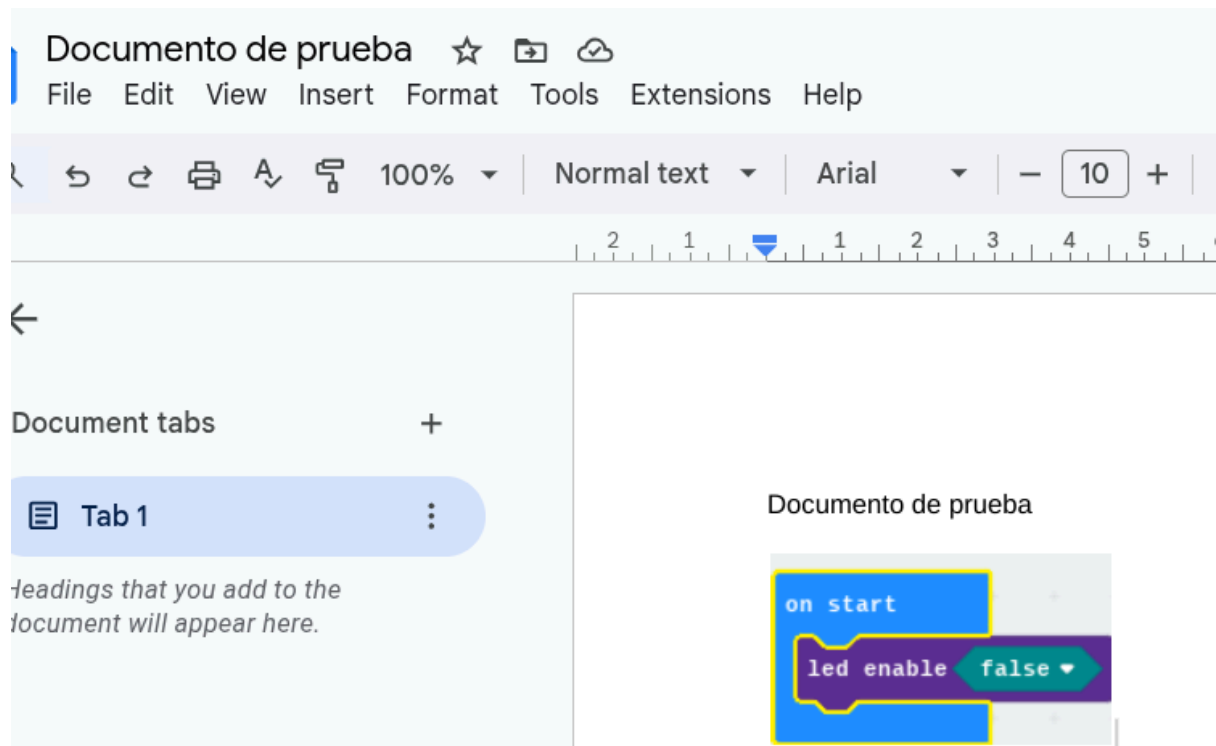
 Ejecutar copia

 Verificar integridad

 Limpiar destino

☒ Todo coincide: no se encontraron diferencias entre origen y destino.

Si el propietario crea un nuevo documento en la carpeta compartida



O más de uno , por ejemplo, otro fichero en otra de las carpetas

My Drive > Mis clases > ej1 ▾ 👤









Type ▾

People ▾

Modified ▾

Source ▾

Name ▾

	style.css	
	sketch.js	
	Leeme	
	index.html	

Si volvemos a verificar la integridad de la copia

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases


La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 **Verificación finalizada con incidencias:**

 Archivo no registrado aún: /Mis clases/DIG/Documento de prueba

 Archivo no registrado aún: /Mis clases/ej1/Leeme

 Puedes volver a lanzar la copia para intentar completar o actualizar los elementos.

Podemos volver a lanzar la copia

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Usando carpeta existente: "Copia de Mis Clases".
 Se han copiado 2 elementos nuevos.

Abrir carpeta en Drive

Ver archivo de progreso

Si hay cambios en los ficheros originales, por ejemplo, he eliminado el Leeme del directorio ej1 y he actualizado el Documento de prueba del directorio DIG

Documento de prueba ☆ 📁 ☁

File Edit View Insert Format Tools Extensions Help

↶ ↷ 🖨 A ↺ 100% ▾ | Normal text ▾ | Arial ▾ | - 10 + | B I U A

2 1 1 2 3 4 5 6 7 8 9

Documento de prueba

```
on start
  led enable false

forever
  analog write pin rojo to encendido
  analog write pin verde to apagado
  pause (ms) 3000
  analog write pin rojo to apagado
  analog write pin verde to encendido
  pause (ms) 3000
  analog write pin rojo to 712
  analog write pin verde to 311
  pause (ms) 3000
```

Si volvemos a lanzar el verificador de integridad de la copia

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

⚠ Verificación finalizada con incidencias:

⚠ Modificado desde la copia: /Mis clases/DIG/Documento de prueba (original: 3/5/2025, 10:18:39, ahora: 3/5/2025, 10:29:13)

⚠ Archivo en destino que ya no está en origen: /Mis clases/ej1/Leeme

 Puedes volver a lanzar la copia para intentar completar o actualizar los elementos.

Si ahora ejecutamos la copia se va a traer la actualización, pero no eliminará del destino el archivo eliminado en el origen

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Usando carpeta existente: "Copia de Mis Clases".
 Se han copiado 1 elementos nuevos.

Abrir carpeta en Drive

Ver archivo de progreso

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:

Copia de Mis Clases


La copia se reanuda automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

Verificación finalizada con incidencias:

 Archivo en destino que ya no está en origen: /Mis clases/ej1/Leeme

 Puedes volver a lanzar la copia para intentar completar o actualizar los elementos.

Si decides eliminar los ficheros del destino que ya no están en el origen, lo puedes hacer manualmente, pero es mejor verificar la integridad, para saber cuales son y después eliminarlos con Limpiar destino. Que te va a preguntar si quieres borrar todos esos ficheros

Una página insertada en n-
pghfxnt3syrrnzhtqqd2bz36b7jpq2oe3akzacq-1lu-
script.googleusercontent.com dice

⚠ ¿Estás seguro de que deseas eliminar los elementos sobrantes del destino?

Cancelar

Aceptar

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:


Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Se han eliminado 1 elementos sobrantes:

- Archivo eliminado del destino: /Mis clases/ej1/Leeme

Si por error eliminas ficheros del destino (o una carpeta entera por poner un ejemplo) Yo he borrado "accidentalmente" la carpeta "CyR" con todo su contenido y no quiero ir a las herramientas del drive para recuperar lo eliminado, sino que me quiero traer de nuevo del origen lo que allí tenía

Mi unidad > Copia de Mis Clases ▾




Tipo ▾

Personas ▾

Modificado ▾

Fuente ▾

Nombre ↑

	DIG
	ej1
	myProject.zip

Si verifico la integridad de mi copia (aunque no es necesario hacer esto primero) solo es por mostrar que se detecta la ausencia de la carpeta

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.


 Ejecutar copia


 Verificar integridad


 Limpiar destino

Verificación finalizada con incidencias:

 Subcarpeta eliminada (en papelera): /Mis clases/CyR

 Modificado desde la copia: /Mis clases/DIG/Documento de prueba (original: 3/5/2025, 11:24:19, ahora: 3/5/2025, 11:24:25)

 Falta subcarpeta en destino: /Mis clases/CyR

 Puedes volver a lanzar la copia para intentar completar o actualizar los elementos.

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Usando carpeta existente: "Copia de Mis Clases".
 Se han copiado 5 elementos nuevos.

Abrir carpeta en Drive

Ver archivo de progreso

Otro ejemplo con "perdida de la carpeta de destino "DIG" y varios ficheros de la carpeta de destino "ej1"

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:

Copia de Mis Clases





La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

Verificación finalizada con incidencias:

-  Subcarpeta eliminada (en papelera): /Mis clases/DIG
-  Falta subcarpeta en destino: /Mis clases/DIG
-  Archivo eliminado (en papelera): /Mis clases/ej1/style.css
-  Archivo eliminado (en papelera): /Mis clases/ej1/sketch.js

 Puedes volver a lanzar la copia para intentar completar o actualizar los elementos.

Como ves te avisa que la pérdida es recuperable ya que están en la papelera, pero vamos a darle a Ejecutar copia para intentar traerlos del origen

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFedw4z

Nombre de la carpeta destino:



Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

 Copia de Mis Clases
Usando carpeta existente: "Copia de Mis Clases".
 Se han copiado 7 elementos nuevos.

Abrir carpeta en Drive

Ver archivo de progreso

Los 7 elementos traídos son los dos ficheros eliminados de la carpeta ej1 del destino, los cuatro ficheros que estaban dentro de la carpeta DIG y la propia carpeta DIG

Si le damos por error a Limpiar Destino y aceptamos el mensaje de eliminar sobrantes, cuando no hay nada que borrar, como nada debe borrarse nos informa de que nada se ha borrado

Copia desde carpeta compartida

ID de la carpeta compartida:

176sQ9DtavDSTmW_T7o1KOH1PXZFdw4z

Nombre de la carpeta destino:

Copia de Mis Clases

La copia se reanudará automáticamente si ya existe contenido.
No se sobrescriben ni se duplican archivos.

 Ejecutar copia

 Verificar integridad

 Limpiar destino

☒ No había elementos sobrantes. Nada se ha eliminado.

Limitaciones

Si el elemento a eliminar de la carpeta de destino ya no estaba en la carpeta origen y lo hemos eliminado por error como sobrante, cuando no queríamos eliminarlo todavía, la única forma de recuperarlo es vía la papelera de reciclaje de nuestro drive y ahí la aplicación no te puede ayudar.

Otra limitación es con cambios que hagamos en la carpeta de destino, por ejemplo, editamos un archivo en la carpeta de destino, la **verificación de integridad** no va a detectar estos cambios posteriores a la copia y **ejecutar copia** tampoco va a hacer nada.

Ten en cuenta que si hacemos cambios en el origen, el verificador los va a detectar y el ejecutar copia va a machacar la copia de destino con los cambios que tengamos en el origen.

Recuerda que este script es para generar una copia de seguridad de una carpeta compartida, manteniendo su estructura, no para hacer un control de versiones de edición en vivo. Siempre debe mandar lo que haya en la carpeta de origen.

También es importante recordar el tema de permisos. Si en el origen añadimos por ejemplo un acceso directo a otro archivo al que el propietario de la carpeta de destino no tiene permisos, la copia del acceso directo se hará en el destino, pero reflejará que no se dispone de permisos para ver dónde apunta el acceso directo.

Mi unidad > Copia de Mis Clases ▾

Tipo ▾

Personas ▾

Modificado ▾

Fuente ▾

Nombre ↑	Propietario
 CyR	 yo
 DIG	 yo
 ej1	 yo
 FlipVideoEditor v2 Distribution 	Solicitar acceso
 myProject.zip	 yo

Si tienes accesos directos a otras carpetas, lo apropiado sería hacer también copia de esas otras carpetas, no mantener los accesos directos a los diferentes orígenes.