```
/***** ========================================
 *  CIERRE MENSUAL HÍBRIDO v1.2 (sin colisiones)
 *  - Unifica lectura de RESUMEN (Solicitado/Modificado del mes)
 *  - Cruza con Google Calendar por ID y por (cliente+piso+fecha)
 *  - Incluye eventos solo en Calendar (marca calendar_only = "Sí")
 *  - Enriquece con BASE_DATOS
 *  - Genera fichero <Mes>_Cierre_<YYYY-MM-DD> en carpeta DEST_FOLDER_ID
 *  - Menú independiente: "Cerrar mes" → "Generar archivo (híbrido)"
 *  - Trigger dedicado __onOpen_menu_cierre (no colisiona con onOpen
existente)
 ========================================*/

/** === CONFIG === */
const CIERRE_CFG = {
  HOJA_RESUMEN: 'Resumen',
  HOJA_BASE: 'BASE_DATOS',
  DEST_FOLDER_ID: '1fvcLjNv2-VTyzN5WRaWyPODndHAHuovV', // 00_RESUMEN DATOS
  ESTADOS_PERMITIDOS: ['solicitado','modificado'], // cambia a
['solicitado'] si quieres
  TZ: 'Europe/Madrid'
};

/** === MENÚ === */
function buildMenuCerrarMes(){
  SpreadsheetApp.getUi()
    .createMenu('Cerrar mes')
    .addItem('Generar archivo (híbrido)', 'cerrarMesHibrido')
    .addToUi();
}

/** Trigger onOpen para menú (no colisiona con otros onOpen) */
function createOnOpenTrigger_CerrarMes(){
  ScriptApp.getProjectTriggers().forEach(t=>{
    if (t.getHandlerFunction &&
t.getHandlerFunction()==='__onOpen_menu_cierre'){
      ScriptApp.deleteTrigger(t);
    }
  });
  ScriptApp.newTrigger('__onOpen_menu_cierre')
```

```
    .forSpreadsheet(SpreadsheetApp.getActive())
    .onOpen()
    .create();
  SpreadsheetApp.getActive().toast('Trigger creado. Al reabrir verás
"Cerrar mes".', 'Cerrar mes', 6);
}
function __onOpen_menu_cierre(){ try{ buildMenuCerrarMes(); }catch(e){
Logger.log(e); } }

/** === HELPERS (prefijo cm_) === */
const cm_tz = ()=> CIERRE_CFG.TZ || (Session.getScriptTimeZone() ||
'Europe/Madrid');
const cm_pad2 = n => (n<10 ? '0'+n : ''+n);
const cm_monthNameEs = m0 =>
(['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septie
mbre','Octubre','Noviembre','Diciembre'][m0] || '');
function cm_todayStr(){ return Utilities.formatDate(new Date(), cm_tz(),
'yyyy-MM-dd'); }
function cm_fmtDMY(d){ return Utilities.formatDate(d, cm_tz(),
'dd/MM/yyyy'); }
function cm_normalizeEventId(id){ return
String(id||'').trim().toLowerCase().replace(/@google\.com$/,''); }
function cm_norm(s){ return String(s ??
'').normalize('NFD').replace(/[\u0300-\u036f]/g,'').toLowerCase().trim(); }
function cm_parseFechaCell(v){
  if (v instanceof Date && !isNaN(v)) return v;
  const s = String(v||'').trim(); if (!s) return null;
  let m = s.match(/^(\d{1,2})[\/\-](\d{1,2})[\/\-](\d{2,4})/);
  if (m){ let [_, dd, mm, yy] = m; if (yy.length===2) yy = '20'+yy; return
new Date(+yy, +mm-1, +dd); }
  m = s.match(/^(\d{4})-(\d{2})-(\d{2})/);
  if (m){ return new Date(+m[1], +m[2]-1, +m[3]); }
  const d = new Date(s); return isNaN(d) ? null : d;
}

/** === RESUMEN: carga Solicitado/Modificado del MES ACTUAL === */
function cm_cargarResumenFiltrado(){
  const ss = SpreadsheetApp.getActive();
  const sh = ss.getSheetByName(CIERRE_CFG.HOJA_RESUMEN);
```

```
  if (!sh) return { mapById:{}, rows:[], byPisoCliente:[] };
  const lastRow = sh.getLastRow(), lastCol = sh.getLastColumn();
  if (lastRow < 2) return { mapById:{}, rows:[], byPisoCliente:[] };

  const H = sh.getRange(1,1,1,lastCol).getDisplayValues()[0].map(String);
  const V = sh.getRange(2,1,lastRow-1,lastCol).getValues();

  const idxExact = (label)=>{ const n = cm_norm(label); for (let
i=0;i<H.length;i++) if (cm_norm(H[i])===n) return i; return -1; };
  const iFecha  = idxExact('fecha');
  const iPiso   = idxExact('piso');
  const iCli    = idxExact('cliente');
  const iEstado = idxExact('estado');
  const iIdEv   = H.findIndex(h => cm_norm(h).includes('id evento'));

  const allowed = new Set(CIERRE_CFG.ESTADOS_PERMITIDOS.map(cm_norm));
  const mapById = {}; const rows = []; const byPisoCliente = [];

  const hoy = new Date(); const y = hoy.getFullYear(), m0 = hoy.getMonth();
  const inThisMonth = (d)=> d && d.getFullYear()===y && d.getMonth()===m0;

  for (let r=0;r<V.length;r++){
    const row = V[r];
    const estado = cm_norm(row[iEstado] || '');
    if (!allowed.has(estado)) continue;
    const fecha = cm_parseFechaCell(row[iFecha]);
    if (!inThisMonth(fecha)) continue;

    const piso = String(row[iPiso]||'').trim();
    const cliente = String(row[iCli]||'').trim();

    const getVal = (label)=>{ const j = idxExact(label); return j>=0 ?
V[r][j] : ''; };

    const obj = {
      _row: r+2,
      fecha, piso, cliente, estado,
      marcaTiempo: getVal('marca de tiempo'),
      precioLimpieza: getVal('precio limpieza'),
```

```
      precioRopa: getVal('precio ropa'),
      total: getVal('total') || getVal('total '),
      facturado: getVal('facturado'),
      observaciones: getVal('observaciones'),
      limpiadora: getVal('limpiadora'),
      correoLim: getVal('correo limpiadora'),
      tipo: getVal('tipo'),
      dias: getVal('días') || getVal('dias'),
      clave: getVal('clave'),
      origen: getVal('origen'),
      uid: getVal('uid')
    };

    rows.push(obj);
    const idRaw = (iIdEv>=0 ? row[iIdEv] : '') || '';
    const idNorm = cm_normalizeEventId(idRaw);
    if (idNorm) mapById[idNorm] = obj;
    byPisoCliente.push({piso, cliente, fechaStr: cm_fmtDMY(fecha), obj});
  }
  return { mapById, rows, byPisoCliente };
}

/** === BASE_DATOS === */
function cm_cargarBaseDatos(){
  const sh =
SpreadsheetApp.getActive().getSheetByName(CIERRE_CFG.HOJA_BASE);
  if (!sh) return [];
  const lastRow = sh.getLastRow(), lastCol = sh.getLastColumn();
  if (lastRow < 2) return [];
  const H = sh.getRange(1,1,1,lastCol).getDisplayValues()[0].map(String);
  const V = sh.getRange(2,1,lastRow-1,lastCol).getValues();
  const idx = {}; H.forEach((h,i)=> idx[cm_norm(h)] = i);
  const G = k => idx[cm_norm(k)];
  return V.map(r => ({
    cliente:   r[G('cliente')] || '',
    nif:       r[G('nif')] || r[G('cif')] || r[G('cif/nif')] || '',
    dirFact:   r[G('dirección facturación')] ||
r[G('direccion_facturacion')] || r[G('direccion')] || '',
    persona:   r[G('persona de contacto')] || '',
```

```
    tlf:       r[G('tlfn de contacto')] || r[G('telefono')] || '',
    email:     r[G('email cliente')] || r[G('email')] || '',
    piso:      r[G('piso')] || '',
    poblacion: r[G('población')] || r[G('poblacion')] || '',
    srvLimp:   r[G('servicio limpieza')] || '',
    precioLimp:r[G('precio limpieza')] || '',
    srvRopa:   r[G('servicio ropa')] || '',
    precioRopa:r[G('precio ropa')] || '',
    activo:    r[G('activo')] || ''
  }));
}
function cm_buscarEnBasePorPisoOCliente(base, piso, cliente){
  if (piso){ const hit = base.find(x => cm_norm(x.piso) === cm_norm(piso));
if (hit) return hit; }
  if (cliente){ const hit = base.find(x => cm_norm(x.cliente) ===
cm_norm(cliente)); if (hit) return hit; }
  return null;
}

/** === MATCH auxiliar por (cliente+piso+fecha) si no hay ID === */
function cm_findResumenByComp(resumenByPisoCliente, cliente, piso,
fechaDate){
  const fechaStr = cm_fmtDMY(fechaDate);
  const clienteN = cm_norm(cliente);
  const pisoN = cm_norm(piso);
  return resumenByPisoCliente.find(x => cm_norm(x.cliente)===clienteN &&
cm_norm(x.piso)===pisoN && x.fechaStr===fechaStr)?.obj || null;
}

/** === PRINCIPAL: genera el archivo de cierre del mes actual === */
function cerrarMesHibrido(){
  const hoy = new Date();
  const y = hoy.getFullYear(), m0 = hoy.getMonth();
  const start = new Date(y, m0, 1, 0,0,0);
  const end   = new Date(y, m0+1, 0, 23,59,59);

  const { mapById, byPisoCliente } = cm_cargarResumenFiltrado();
  const base = cm_cargarBaseDatos();
```

```
  const headers = [
    'Calendario','Título','Inicio','Fin','Todo el
día','Descripción','Ubicación','Creador','ID Evento',
    'Marca de Tiempo','Piso','Fecha (Origen)','Cliente (Origen)','Precio
Limpieza (Origen)','Precio Ropa (Origen)','Total (Origen)',
    'Estado (Origen)','Facturado','Observaciones (Origen)','Limpiadora
(Origen)','Correo Limpiadora (Origen)',
    'Tipo','Días','Clave','Origen','UID',
    'Cliente','CIF/NIF','Dirección Facturación','Persona de contacto','Tlfn
de contacto','Email cliente','Población',
    'Servicio Limpieza','Precio Limpieza','Servicio Ropa','Precio
Ropa','Activo',
    'calendar_only' // NUEVA bandera
  ];

  const rowsOut = [];

  // Recorremos TODOS los calendarios
  CalendarApp.getAllCalendars().forEach(cal => {
    cal.getEvents(start, end).forEach(ev => {
      const idRaw  = ev.getId();
      const idNorm = cm_normalizeEventId(idRaw);

      // 1) Intentar por ID → RESUMEN
      let r = mapById[idNorm] || null;

      // 2) Si no hay, intentar por (cliente+piso+fecha)
      if (!r){
        // inferir cliente y piso desde el título con BASE_DATOS
        const title = String(ev.getTitle()||'');
        const likelyBase = base.find(x =>
title.toLowerCase().includes(String(x.piso||'').toLowerCase()))
                          || base.find(x =>
title.toLowerCase().includes(String(x.cliente||'').toLowerCase()))
                          || null;
        if (likelyBase){
          r = cm_findResumenByComp(byPisoCliente, likelyBase.cliente,
likelyBase.piso, ev.getStartTime());
        }
```

```
      }

      const maestro = r ? cm_buscarEnBasePorPisoOCliente(base, r.piso,
r.cliente)
                        : (function(){
                            const t =
String(ev.getTitle()||'').toLowerCase();
                            return base.find(x => x.piso &&
t.includes(String(x.piso).toLowerCase()))
                                || base.find(x => x.cliente &&
t.includes(String(x.cliente).toLowerCase()))
                                || null;
                          })();

      const creators = (ev.getCreators && ev.getCreators()) ?
ev.getCreators().join(', ') : '';

      rowsOut.push([
        cal.getName(),
        ev.getTitle(),
        ev.getStartTime(),
        ev.getEndTime(),
        ev.isAllDayEvent(),
        ev.getDescription(),
        ev.getLocation(),
        creators,
        idRaw,
        r ? r.marcaTiempo : '',
        r ? r.piso : '',
        r ? cm_fmtDMY(r.fecha) : '',
        r ? r.cliente : '',
        r ? r.precioLimpieza : '',
        r ? r.precioRopa : '',
        r ? r.total : '',
        r ? cm_capitalize(r.estado) : '',
        r ? r.facturado : '',
        r ? r.observaciones : '',
        r ? r.limpiadora : '',
        r ? r.correoLim : '',
```

```
        r ? r.tipo : '',
        r ? r.dias : '',
        r ? r.clave : '',
        r ? r.origen : '',
        r ? r.uid : '',
        maestro ? maestro.cliente : '',
        maestro ? maestro.nif : '',
        maestro ? maestro.dirFact : '',
        maestro ? maestro.persona : '',
        maestro ? maestro.tlf : '',
        maestro ? maestro.email : '',
        maestro ? maestro.poblacion : '',
        maestro ? maestro.srvLimp : '',
        maestro ? maestro.precioLimp : '',
        maestro ? maestro.srvRopa : '',
        maestro ? maestro.precioRopa : '',
        maestro ? maestro.activo : '',
        r ? '' : 'Sí' // calendar_only
      ]);
    });
  });

  if (!rowsOut.length){
    SpreadsheetApp.getActive().toast('No hay eventos publicados este mes.',
'Cerrar mes', 6);
    return;
  }

  const fileName = `${cm_monthNameEs(m0)}_Cierre_${cm_todayStr()}`;
  const ssNew = SpreadsheetApp.create(fileName);
  const file = DriveApp.getFileById(ssNew.getId());
  try{
    const folder = DriveApp.getFolderById(CIERRE_CFG.DEST_FOLDER_ID);
    folder.addFile(file);
    try{ DriveApp.getRootFolder().removeFile(file); }catch(_){/* ignore */}
  }catch(e){ Logger.log('⚠ No se pudo mover el archivo: '+e); }

  const sh = ssNew.getActiveSheet();
  sh.setName('Cierre_mes');
```

```
  sh.getRange(1,1,1,headers.length).setValues([headers]);
  sh.getRange(2,1,rowsOut.length,headers.length).setValues(rowsOut);
  sh.setFrozenRows(1);
  sh.autoResizeColumns(1, headers.length);
  SpreadsheetApp.getActive().toast(`Creado: ${fileName} · ${rowsOut.length} filas`, 'Cerrar mes', 6);
  Logger.log(`✅ Cierre generado: ${fileName} (${rowsOut.length} filas).`);
}

function cm_capitalize(s){ s = String(s||'').toLowerCase(); return s ? s[0].toUpperCase()+s.slice(1) : s; }
```