```
// ============================================================
// PAI RISARALDA · Sistema de Alertas de Inventario de Vacunas
// Google Apps Script — vinculado al Google Sheet del inventario
// Secretaría de Salud de Pereira · 2026
// ============================================================
// INSTALACIÓN:
// 1. En el Google Sheet → Extensiones → Apps Script
// 2. Pega este código completo
// 3. Ejecuta la función "instalarTriggers" UNA sola vez
// 4. Autoriza los permisos que solicite
// 5. Listo. El sistema corre automáticamente.
// ============================================================

// — CONFIGURACIÓN ————————————————————————————

const SHEET_ID = "1vGm2hVwZ1UwuoyEynZtA-AVhlMdDUNtD1NkyvXoDnxY";

const CONFIG = {

  EMAIL_ADMIN: "risaraldapaiweb@gmail.com",      // ← Cambia esto

  NOMBRE_HOJA_HISTORIAL: "🗄️ Historial",

  NOMBRE_HOJA_INFORME:   "📊 Informe Semanal",

  NOMBRE_HOJA_ALERTAS:   "🚨 Alertas",

  HOJAS_EXCLUIDAS: ["Seguimiento", "Consolidado", "Inventario Gral",

                    "Espacio en Blanco", "🗄️ Historial",

                    "📊 Informe Semanal", "🚨 Alertas"],

  MAX_SNAPSHOTS: 12,   // semanas de historial a conservar

  UMBRAL_ALERTA_DIAS: 7,  // días sin cambio para disparar alerta amarilla

  UMBRAL_CRITICO_DIAS: 14, // días sin cambio para alerta roja
```

```
};

// — COLORES ————————————————————————————————————————

const COLOR = {
  VERDE_FONDO:    "#e1f5ee", VERDE_TEXTO:    "#085041", VERDE_BORDE:    "#1d9e75",
  AMARILLO_FONDO: "#faeeda", AMARILLO_TEXTO: "#633806", AMARILLO_BORDE:
"#ef9f27",
  ROJO_FONDO:     "#fcebeb", ROJO_TEXTO:     "#791f1f", ROJO_BORDE:     "#e24b4a",
  GRIS_FONDO:     "#f1efe8", GRIS_TEXTO:     "#444441",
  AZUL_FONDO:     "#e6f1fb", AZUL_TEXTO:     "#042c53", AZUL_BORDE:     "#185fa5",
  ENCABEZADO:     "#1a1a18", ENCABEZADO_TXT: "#ffffff",
  SUBENCABEZADO:  "#f5f5f0", SUBENC_TXT:     "#555555",
  BLANCO:         "#ffffff", BORDE_SUAVE:    "#e0e0d8",
};

//
═══════════════════════════════════════════════════════════════════════════
═══

// 1. INSTALACIÓN DE TRIGGERS (ejecutar solo una vez)
//
═══════════════════════════════════════════════════════════════════════════
═══

function instalarTriggers() {
  ScriptApp.getProjectTriggers().forEach(t => ScriptApp.deleteTrigger(t));

  ScriptApp.newTrigger("tomarSnapshot")
    .timeBased()
    .onWeekDay(ScriptApp.WeekDay.MONDAY)
    .atHour(11)
    .create();
```

```
  ScriptApp.newTrigger("generarInformeCompleto")
    .timeBased()
    .onWeekDay(ScriptApp.WeekDay.TUESDAY)
    .atHour(8)
    .create();

  Logger.log("✅ Triggers instalados.");
  Logger.log("• Snapshot: cada lunes a las 11 AM (hora Colombia)");
  Logger.log("• Informe: cada martes a las 8 AM (hora Colombia)");
}

//
═══════════════════════════════════════════════════════════════════════
═══
// 2. CAPTURA DE SNAPSHOT SEMANAL
//
═══════════════════════════════════════════════════════════════════════
═══
function tomarSnapshot() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const datos = leerInventarioActual(ss);
  const timestamp = new Date().toISOString();

  const snapshot = {
    ts: timestamp,
    fecha_humana: Utilities.formatDate(new Date(), "America/Bogota", "dd/MM/yyyy HH:mm"),
    datos: datos
  };

  guardarSnapshot(ss, snapshot);
```

```
  Logger.log("Snapshot tomado: " + snapshot.fecha_humana);
  return snapshot;
}

function leerInventarioActual(ss) {
  const hojas = ss.getSheets().filter(h =>
!CONFIG.HOJAS_EXCLUIDAS.includes(h.getName()));
  const resultado = {};

  hojas.forEach(hoja => {
    const nombre = hoja.getName();
    const valores = hoja.getDataRange().getValues();

    // Buscar nombre de institución y fecha
    let institución = null, fecha = null;
    for (let r = 0; r < Math.min(8, valores.length); r++) {
      const fila = valores[r];
      for (let c = 0; c < fila.length; c++) {
        const cel = String(fila[c]).trim();
        if (cel === "Institución" && !institución) {
          for (let nc = c + 1; nc < fila.length; nc++) {
            const v = String(fila[nc]).trim();
            if (v && v !== "nan" && v !== "Institución" && v !== "-") {
              institución = v; break;
            }
          }
        }
        if (cel === "Fecha Actualización" && !fecha) {
          for (let nc = c + 1; nc < fila.length; nc++) {
```

```
        const v = fila[nc];
        if (v && String(v).trim() !== "-" && String(v).trim() !== "") {
          fecha = v instanceof Date ? Utilities.formatDate(v, "America/Bogota", "dd/MM/yyyy") :
String(v).trim();
          break;
        }
       }
     }
    }
  }

  // Encontrar fila de encabezados de datos (busca "Insumo")
  let dataStart = -1;
  for (let r = 0; r < valores.length; r++) {
   if (valores[r].some(c => String(c).trim() === "Insumo")) {
    dataStart = r + 1; break;
   }
  }

  // Leer vacunas e insumos
  const vacunas = {};
  if (dataStart >= 0) {
   for (let r = dataStart; r < valores.length; r++) {
    const fila = valores[r];
    const insumo = String(fila[2] || "").trim();
    if (!insumo || insumo === "nan") continue;
    let qty = 0;
    [3, 6, 9].forEach(ci => {
     const v = parseFloat(String(fila[ci] || "0").replace(",", "."));
```

```
        if (!isNaN(v)) qty += v;
       });
       vacunas[insumo] = Math.round(qty);
     }
    }

    resultado[nombre] = {
     nombre: institución || nombre,
     fecha: fecha || "",
     vacunas: vacunas
    };
  });

  return resultado;
}

function guardarSnapshot(ss, snapshot) {
  let hojaHistorial = ss.getSheetByName(CONFIG.NOMBRE_HOJA_HISTORIAL);
  if (!hojaHistorial) {
    hojaHistorial = ss.insertSheet(CONFIG.NOMBRE_HOJA_HISTORIAL);
    hojaHistorial.hideSheet();
    hojaHistorial.getRange("A1").setValue("timestamp");
    hojaHistorial.getRange("B1").setValue("fecha_humana");
    hojaHistorial.getRange("C1").setValue("datos_json");
    hojaHistorial.getRange("1:1").setFontWeight("bold");
  }

  const ultimaFila = hojaHistorial.getLastRow();
  hojaHistorial.getRange(ultimaFila + 1, 1, 1, 3).setValues([
```

```
    [snapshot.ts, snapshot.fecha_humana, JSON.stringify(snapshot.datos)]
  ]);

  // Limpiar snapshots viejos si excede el máximo
  const totalFilas = hojaHistorial.getLastRow() - 1; // sin encabezado
  if (totalFilas > CONFIG.MAX_SNAPSHOTS) {
    const sobran = totalFilas - CONFIG.MAX_SNAPSHOTS;
    hojaHistorial.deleteRows(2, sobran);
  }
}

function obtenerHistorial(ss) {
  const hojaHistorial = ss.getSheetByName(CONFIG.NOMBRE_HOJA_HISTORIAL);
  if (!hojaHistorial || hojaHistorial.getLastRow() < 2) return [];

  const datos = hojaHistorial.getRange(2, 1, hojaHistorial.getLastRow() - 1, 3).getValues();
  return datos.map(fila => ({
    ts: fila[0],
    fecha_humana: fila[1],
    datos: JSON.parse(fila[2])
  })).reverse(); // más reciente primero
}

//
═══════════════════════════════════════════════════════════════════════
═══
// 3. ANÁLISIS COMPARATIVO
//
═══════════════════════════════════════════════════════════════════════
═══
```

```
function analizarCambios(snapshotActual, snapshotAnterior) {
  const curr = snapshotActual.datos;
  const prev = snapshotAnterior ? snapshotAnterior.datos : {};
  const todas_ips = new Set([...Object.keys(curr), ...Object.keys(prev)]);
  const resultado = [];

  todas_ips.forEach(key => {
    const c = curr[key] || {};
    const p = prev[key] || {};
    const nombre = c.nombre || p.nombre || key;
    const cVacs = c.vacunas || {};
    const pVacs = p.vacunas || {};

    const todosInsumos = new Set([...Object.keys(cVacs), ...Object.keys(pVacs)]);
    let n_changed = 0, consumidas = 0, repuestas = 0;
    const insumos = [];

    todosInsumos.forEach(ins => {
      const cv = cVacs[ins] || 0;
      const pv = pVacs[ins] || 0;
      const diff = cv - pv;
      const changed = cv !== pv;
      if (changed) {
        n_changed++;
        if (diff < 0) consumidas += Math.abs(diff);
        if (diff > 0) repuestas += diff;
      }
      insumos.push({
        nombre: ins,
```

```
    prev: pv,
    curr: cv,
    diff: diff,
    changed: changed,
    dias: changed ? 0 : 7
  });
});

// Ordenar: modificados primero, luego por cantidad desc
insumos.sort((a, b) => {
  if (a.changed !== b.changed) return b.changed ? 1 : -1;
  return b.curr - a.curr;
});

const fechaCambio = c.fecha !== p.fecha;
let estado;
if (!snapshotAnterior) {
  estado = "primer_corte";
} else if (n_changed > 0) {
  estado = "ok";
} else if (fechaCambio && n_changed === 0) {
  estado = "solo_fecha";
} else {
  const total = Object.values(cVacs).reduce((s, v) => s + v, 0);
  estado = total === 0 ? "sin_datos" : "no_actualizo";
}

resultado.push({
  key, nombre,
```

```
      fecha_prev: p.fecha || "",
      fecha_curr: c.fecha || "",
      estado,
      n_changed,
      total_insumos: todosInsumos.size,
      consumidas,
      repuestas,
      insumos
    });
  });

  // Ordenar: problemas primero
  const ordenEstado = { sin_datos: 0, solo_fecha: 1, no_actualizo: 2, primer_corte: 3, ok: 4 };
  resultado.sort((a, b) => {
    const oa = ordenEstado[a.estado] ?? 9, ob = ordenEstado[b.estado] ?? 9;
    if (oa !== ob) return oa - ob;
    return a.nombre.localeCompare(b.nombre, "es");
  });

  return resultado;
}

//
═══════════════════════════════════════════════════════════════════════
// 4. GENERACIÓN DEL INFORME EN GOOGLE SHEETS
//
═══════════════════════════════════════════════════════════════════════
function generarInformeCompleto() {
```

```
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const historial = obtenerHistorial(ss);

  if (historial.length === 0) {
   Logger.log("⚠️ No hay snapshots previos. Tomando primer snapshot ahora...");
   tomarSnapshot();
   return;
  }

  const snapshotActual = historial[0];
  const snapshotAnterior = historial.length > 1 ? historial[1] : null;
  const analisis = analizarCambios(snapshotActual, snapshotAnterior);

  generarHojaInforme(ss, analisis, snapshotActual, snapshotAnterior);
  generarHojaAlertas(ss, analisis, snapshotActual);

  // Enviar email siempre (no solo cuando hay problemas)
  enviarEmailAlerta(analisis, snapshotActual, snapshotAnterior);

  Logger.log("Informe generado: " + new Date());
}

function generarHojaInforme(ss, analisis, snapAct, snapPrev) {
  // Eliminar hoja previa y crear nueva
  let hoja = ss.getSheetByName(CONFIG.NOMBRE_HOJA_INFORME);
  if (hoja) ss.deleteSheet(hoja);
  hoja = ss.insertSheet(CONFIG.NOMBRE_HOJA_INFORME);
  ss.setActiveSheet(hoja);
  ss.moveActiveSheet(1); // Mover al frente
```

```
const fechaInforme = Utilities.formatDate(new Date(), "America/Bogota", "dd/MM/yyyy");
const periodoStr = snapPrev
 ? `Comparativo: ${snapPrev.fecha_humana} → ${snapAct.fecha_humana}`
 : `Primer corte: ${snapAct.fecha_humana}`;

// — ENCABEZADO —
let fila = 1;
hoja.getRange(fila, 1, 1, 10).merge()
 .setValue("📊 INFORME SEMANAL DE INVENTARIO DE VACUNAS · PAI RISARALDA 2026")
 .setBackground(COLOR.ENCABEZADO).setFontColor(COLOR.ENCABEZADO_TXT)
 .setFontSize(14).setFontWeight("bold").setHorizontalAlignment("center")
 .setVerticalAlignment("middle");
hoja.setRowHeight(fila, 42);
fila++;

hoja.getRange(fila, 1, 1, 10).merge()
 .setValue("Secretaría de Salud de Pereira · " + periodoStr)
 .setBackground(COLOR.AZUL_FONDO).setFontColor(COLOR.AZUL_TEXTO)
 .setFontSize(11).setHorizontalAlignment("center").setVerticalAlignment("middle");
hoja.setRowHeight(fila, 26);
fila++;

hoja.getRange(fila, 1, 1, 10).merge()
 .setValue("Generado el " + fechaInforme)
 .setFontColor("#888888").setFontSize(10).setHorizontalAlignment("center")
 .setVerticalAlignment("middle").setFontStyle("italic");
hoja.setRowHeight(fila, 22);
fila += 2;
```

```
// — RESUMEN ESTADÍSTICO —
const totalIPS = analisis.length;
const ok = analisis.filter(i => i.estado === "ok").length;
const soloFecha = analisis.filter(i => i.estado === "solo_fecha").length;
const sinDatos = analisis.filter(i => i.estado === "sin_datos").length;
const noActualizo = analisis.filter(i => i.estado === "no_actualizo").length;
const problemas = soloFecha + sinDatos + noActualizo;
const pct = Math.round(ok / totalIPS * 100);
const totalConsumidas = analisis.reduce((s, i) => s + i.consumidas, 0);
const totalRepuestas = analisis.reduce((s, i) => s + i.repuestas, 0);

hoja.getRange(fila, 1, 1, 10).merge()
 .setValue("RESUMEN DEL CORTE")
 .setBackground(COLOR.SUBENCABEZADO).setFontColor(COLOR.SUBENC_TXT)
 .setFontSize(10).setFontWeight("bold").setHorizontalAlignment("left")

fila++;

// Tarjetas de resumen
const tarjetas = [
 ["Total IPS", totalIPS, COLOR.AZUL_FONDO, COLOR.AZUL_TEXTO],
 ["Actualizaron ✅", ok, COLOR.VERDE_FONDO, COLOR.VERDE_TEXTO],
 ["Solo la fecha ⚠️", soloFecha, COLOR.AMARILLO_FONDO, COLOR.AMARILLO_TEXTO],
 ["Sin datos ⛔", sinDatos, COLOR.ROJO_FONDO, COLOR.ROJO_TEXTO],
 ["No actualizaron 🔴", noActualizo, COLOR.ROJO_FONDO, COLOR.ROJO_TEXTO],
];

tarjetas.forEach((t, i) => {
```

```
  const col = i * 2 + 1;
  hoja.getRange(fila, col, 1, 2).merge()
    .setValue(t[0]).setBackground(t[2]).setFontColor(t[3])
    .setFontSize(10).setFontWeight("bold").setHorizontalAlignment("center")
    .setVerticalAlignment("middle");
  hoja.getRange(fila + 1, col, 1, 2).merge()
    .setValue(t[1]).setBackground(t[2]).setFontColor(t[3])
    .setFontSize(24).setFontWeight("bold").setHorizontalAlignment("center")
    .setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 22);
  hoja.setRowHeight(fila + 1, 40);
 });
 fila += 3;

 // Fila de métricas adicionales
 hoja.getRange(fila, 1, 1, 10).merge()
   .setValue(`Tasa de cumplimiento: ${pct}%  |  Dosis consumidas esta semana:
${totalConsumidas.toLocaleString("es-CO")}  |  Dosis repuestas:
${totalRepuestas.toLocaleString("es-CO")}`)
   .setBackground(COLOR.GRIS_FONDO).setFontColor(COLOR.GRIS_TEXTO)
   .setFontSize(10).setHorizontalAlignment("center").setVerticalAlignment("middle");
 hoja.setRowHeight(fila, 26);
 fila += 2;

 // ── SECCIÓN DE ALERTAS ──
 if (problemas > 0) {
   hoja.getRange(fila, 1, 1, 10).merge()
     .setValue("⚠️  IPS CON PROBLEMAS DE ACTUALIZACIÓN")
     .setBackground(COLOR.ROJO_FONDO).setFontColor(COLOR.ROJO_TEXTO)
```

```
    .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("center")
    .setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 30);
  fila++;

  const encAlerta = ["IPS", "Estado", "Fecha anterior", "Fecha actual", "Datos modificados", "Observación"];
  hoja.getRange(fila, 1, 1, encAlerta.length).setValues([encAlerta])
    .setBackground(COLOR.ROJO_TEXTO).setFontColor(COLOR.BLANCO)
    .setFontWeight("bold").setFontSize(10);
  hoja.setRowHeight(fila, 22);
  fila++;

  analisis.filter(i => ["solo_fecha","sin_datos","no_actualizo"].includes(i.estado)).forEach(ips => {
    const labels = { solo_fecha:"Solo cambió la fecha", sin_datos:"Sin datos registrados", no_actualizo:"No actualizó" };
    const obs = ips.estado === "solo_fecha"
      ? `Fecha actualizada pero ${ips.total_insumos} insumos sin cambio`
      : ips.estado === "sin_datos"
      ? "Todos los valores en cero — sin actividad"
      : `${ips.total_insumos} insumos sin modificar`;

    hoja.getRange(fila, 1, 1, 6).setValues([[
      ips.nombre, labels[ips.estado], ips.fecha_prev || "—", ips.fecha_curr || "—",
      `${ips.n_changed}/${ips.total_insumos}`, obs
    ]]).setBackground(COLOR.ROJO_FONDO).setFontColor(COLOR.ROJO_TEXTO)
      .setFontSize(10).setBorder(true, true, true, true, true, true, COLOR.ROJO_BORDE, SpreadsheetApp.BorderStyle.SOLID);
    hoja.setRowHeight(fila, 22);
```

```
    fila++;
  });
  fila++;
}

// — TABLA PRINCIPAL: TODAS LAS IPS —
hoja.getRange(fila, 1, 1, 10).merge()
  .setValue("DETALLE POR IPS — SEMANA " + (snapAct.fecha_humana || ""))
  .setBackground(COLOR.ENCABEZADO).setFontColor(COLOR.ENCABEZADO_TXT)
  .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("center")
  .setVerticalAlignment("middle");
hoja.setRowHeight(fila, 30);
fila++;

const encDetalle = ["IPS", "Estado", "Fecha ant.", "Fecha act.", "Insumos mod.", "Dosis consumidas", "Dosis repuestas", "Sin movimiento", "Fecha cambió", "Observaciones"];
hoja.getRange(fila, 1, 1, encDetalle.length).setValues([encDetalle])
  .setBackground(COLOR.ENCABEZADO).setFontColor(COLOR.BLANCO)
  .setFontWeight("bold").setFontSize(10);
hoja.setRowHeight(fila, 22);
fila++;

const estadoLabels = {
  ok: "✅ Actualizó", solo_fecha: "⚠️ Solo fecha",
  sin_datos: "⛔ Sin datos", no_actualizo: "🔴 No actualizó", primer_corte: "🆕 Primer corte"
};
const estadoColores = {
  ok: [COLOR.VERDE_FONDO, COLOR.VERDE_TEXTO],
```

```
  solo_fecha: [COLOR.AMARILLO_FONDO, COLOR.AMARILLO_TEXTO],
  sin_datos: [COLOR.ROJO_FONDO, COLOR.ROJO_TEXTO],
  no_actualizo: [COLOR.ROJO_FONDO, COLOR.ROJO_TEXTO],
  primer_corte: [COLOR.AZUL_FONDO, COLOR.AZUL_TEXTO]
 };

 analisis.forEach((ips, idx) => {
  const [bgRow, textRow] = estadoColores[ips.estado] || [COLOR.BLANCO,
COLOR.GRIS_TEXTO];
  const bgAlt = idx % 2 === 0 ? "#fafaf6" : COLOR.BLANCO;
  const sinMov = ips.insumos.filter(i => !i.changed && i.curr > 0).length;
  const obs = ips.estado === "solo_fecha"
   ? "⚠ Cambió la fecha pero sin datos nuevos"
   : ips.estado === "sin_datos"
   ? "⛔ Sin ningún registro activo"
   : ips.estado === "no_actualizo"
   ? "🔴 No hubo cambios en datos ni fecha"
   : sinMov > 0
   ? `${sinMov} insumos con stock sin cambio`
   : "";

  hoja.getRange(fila, 1, 1, 10).setValues([[
   ips.nombre,
   estadoLabels[ips.estado] || ips.estado,
   ips.fecha_prev || "—",
   ips.fecha_curr || "—",
   `${ips.n_changed}/${ips.total_insumos}`,
   ips.consumidas > 0 ? `-${ips.consumidas.toLocaleString("es-CO")}` : "—",
   ips.repuestas > 0 ? `+${ips.repuestas.toLocaleString("es-CO")}` : "—",
```

```
    sinMov,
    ips.fecha_prev !== ips.fecha_curr ? "Sí" : "No",
    obs
   ]]);

   const rango = hoja.getRange(fila, 1, 1, 10);
   if (ips.estado !== "ok" && ips.estado !== "primer_corte") {
    rango.setBackground(bgRow).setFontColor(textRow);
   } else {
    rango.setBackground(bgAlt).setFontColor(COLOR.GRIS_TEXTO);
   }
   rango.setFontSize(10);
   hoja.setRowHeight(fila, 20);
   fila++;
  });
  fila += 2;

  // — SECCIÓN DE MOVIMIENTOS POR BIOLÓGICO —
  hoja.getRange(fila, 1, 1, 10).merge()
   .setValue("MOVIMIENTO POR BIOLÓGICO E INSUMO (insumos con cambios)")
   .setBackground(COLOR.ENCABEZADO).setFontColor(COLOR.BLANCO)
   .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("center")
   .setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 30);
  fila++;

  const encBio = ["IPS", "Biológico / Insumo", "Semana anterior", "Semana actual",
"Diferencia", "Tipo de movimiento", "Días sin cambio"];
  hoja.getRange(fila, 1, 1, encBio.length).setValues([encBio])
```

```
 .setBackground(COLOR.AZUL_TEXTO).setFontColor(COLOR.BLANCO)
 .setFontWeight("bold").setFontSize(10);
hoja.setRowHeight(fila, 22);
fila++;

analisis.forEach(ips => {
 ips.insumos.filter(i => i.changed || i.curr > 0).slice(0, 30).forEach((ins, idx) => {
  const tipo = ins.diff > 0 ? " 📦 Reposición" : ins.diff < 0 ? " 💉 Consumo" : " ➖ Sin cambio";
  const bgFila = !ins.changed && ins.curr > 0
   ? COLOR.AMARILLO_FONDO
   : ins.diff > 0
   ? COLOR.VERDE_FONDO
   : ins.diff < 0
   ? "#fff0f0"
   : idx % 2 === 0 ? "#fafaf6" : COLOR.BLANCO;

  hoja.getRange(fila, 1, 1, 7).setValues([[
   ips.nombre,
   ins.nombre,
   ins.prev,
   ins.curr,
   ins.diff > 0 ? `+${ins.diff}` : ins.diff === 0 ? "0" : ins.diff,
   tipo,
   ins.dias
  ]]);
  hoja.getRange(fila, 1, 1, 7).setBackground(bgFila).setFontSize(10);
  hoja.setRowHeight(fila, 19);
  fila++;
 });
```

```
  });

  // — FORMATO DE COLUMNAS —
  hoja.setColumnWidth(1, 180);  // IPS
  hoja.setColumnWidth(2, 130);  // Estado
  hoja.setColumnWidth(3, 110);  // Fecha ant
  hoja.setColumnWidth(4, 110);  // Fecha act
  hoja.setColumnWidth(5, 100);  // Insumos
  hoja.setColumnWidth(6, 120);  // Consumidas
  hoja.setColumnWidth(7, 110);  // Repuestas
  hoja.setColumnWidth(8, 110);  // Sin mov
  hoja.setColumnWidth(9, 90);   // Fecha cambió
  hoja.setColumnWidth(10, 280); // Obs

  // Congelar primera fila (desplazamiento)
  hoja.setFrozenRows(0);

  Logger.log("Hoja de informe generada.");
}

//
═══════════════════════════════════════════════════════════════════════
═══
// 5. HOJA DE ALERTAS (vista simplificada para administradores)
//
═══════════════════════════════════════════════════════════════════════
═══
function generarHojaAlertas(ss, analisis, snapshotActual) {
  let hoja = ss.getSheetByName(CONFIG.NOMBRE_HOJA_ALERTAS);
  if (hoja) ss.deleteSheet(hoja);
```

```
  hoja = ss.insertSheet(CONFIG.NOMBRE_HOJA_ALERTAS);
  ss.moveActiveSheet(2);

  const fechaHoy = Utilities.formatDate(new Date(), "America/Bogota", "dd/MM/yyyy
HH:mm");
  let fila = 1;

  // Título
  hoja.getRange(fila, 1, 1, 6).merge()
    .setValue(" 🚨 ALERTAS DE INVENTARIO PAI · " + fechaHoy)
    .setBackground(COLOR.ENCABEZADO).setFontColor(COLOR.BLANCO)
    .setFontSize(13).setFontWeight("bold").setHorizontalAlignment("center")
    .setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 38);
  fila += 2;

  // – ALERTAS CRÍTICAS (no actualizaron) –
  const criticos = analisis.filter(i =>
["sin_datos","no_actualizo","solo_fecha"].includes(i.estado));

  hoja.getRange(fila, 1, 1, 6).merge()
    .setValue(` 🔴 CRÍTICO: ${criticos.length} IPS sin actualización real `)
    .setBackground(COLOR.ROJO_FONDO).setFontColor(COLOR.ROJO_TEXTO)
    .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("left")
    .setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 28);
  fila++;

  if (criticos.length === 0) {
```

```
  hoja.getRange(fila, 1, 1, 6).merge()
    .setValue("✅ Todas las IPS actualizaron correctamente esta semana.")
    .setBackground(COLOR.VERDE_FONDO).setFontColor(COLOR.VERDE_TEXTO)
    .setFontSize(11).setHorizontalAlignment("center").setVerticalAlignment("middle");
  hoja.setRowHeight(fila, 28);
  fila++;
 } else {
  const encCrit = ["IPS", "Estado", "Última fecha", "Insumos sin cambio", "Total dosis",
"Acción recomendada"];
  hoja.getRange(fila, 1, 1, encCrit.length).setValues([encCrit])
    .setBackground(COLOR.ROJO_TEXTO).setFontColor(COLOR.BLANCO)
    .setFontWeight("bold").setFontSize(10);
  hoja.setRowHeight(fila, 20);
  fila++;

  criticos.forEach(ips => {
    const totalDosis = ips.insumos.reduce((s, i) => s + i.curr, 0);
    const accion = ips.estado === "solo_fecha"
      ? "Contactar IPS — solo cambió la fecha"
      : ips.estado === "sin_datos"
      ? "Verificar si IPS está activa"
      : "Contactar responsable inmediatamente";
    hoja.getRange(fila, 1, 1, 6).setValues([[
      ips.nombre, ips.estado.replace("_"," ").toUpperCase(),
      ips.fecha_curr || "Sin fecha", ips.total_insumos - ips.n_changed,
      totalDosis, accion
    ]]).setBackground(COLOR.ROJO_FONDO).setFontColor(COLOR.ROJO_TEXTO).setFontSiz
e(10);
    hoja.setRowHeight(fila, 20);
```

```
    fila++;
  });
 }
 fila += 2;

 // – INSUMOS CON STOCK SIN MODIFICAR (≥7 días) –
 hoja.getRange(fila, 1, 1, 6).merge()
  .setValue(" ⚠️ INSUMOS CON STOCK SIN MODIFICAR ESTA SEMANA")
  .setBackground(COLOR.AMARILLO_FONDO).setFontColor(COLOR.AMARILLO_TEXTO)
  .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("left")
  .setVerticalAlignment("middle");
 hoja.setRowHeight(fila, 28);
 fila++;

 const encSinMov = ["IPS", "Biológico / Insumo", "Días sin cambio", "Stock actual", "Semana anterior", "Observación"];
 hoja.getRange(fila, 1, 1, encSinMov.length).setValues([encSinMov])
  .setBackground(COLOR.AMARILLO_TEXTO).setFontColor(COLOR.BLANCO)
  .setFontWeight("bold").setFontSize(10);
 hoja.setRowHeight(fila, 20);
 fila++;

 let contadorSinMov = 0;
 analisis.forEach(ips => {
  ips.insumos.filter(i => !i.changed && i.curr > 0 && i.dias >= 7).forEach(ins => {
   const obs = ins.curr > 100 ? "Stock alto — verificar si se consumió" : "Stock normal sin movimiento";
   hoja.getRange(fila, 1, 1, 6).setValues([[
    ips.nombre, ins.nombre, ins.dias, ins.curr, ins.prev, obs
```

```
    ]]).setBackground(COLOR.AMARILLO_FONDO).setFontColor(COLOR.AMARILLO_TEXTO).setFontSize(10);
    hoja.setRowHeight(fila, 19);
    fila++;
    contadorSinMov++;
  });
 });

 if (contadorSinMov === 0) {
  hoja.getRange(fila, 1, 1, 6).merge()
   .setValue("✅ Todos los insumos con stock fueron revisados esta semana.")
   .setBackground(COLOR.VERDE_FONDO).setFontColor(COLOR.VERDE_TEXTO)
   .setFontSize(10).setHorizontalAlignment("center");
  fila++;
 }
 fila += 2;

 // – IPS QUE SÍ ACTUALIZARON –
 const correctas = analisis.filter(i => i.estado === "ok");
 hoja.getRange(fila, 1, 1, 6).merge()
  .setValue(`✅ IPS QUE ACTUALIZARON CORRECTAMENTE (${correctas.length})`)
  .setBackground(COLOR.VERDE_FONDO).setFontColor(COLOR.VERDE_TEXTO)
  .setFontSize(11).setFontWeight("bold").setHorizontalAlignment("left")
  .setVerticalAlignment("middle");
 hoja.setRowHeight(fila, 28);
 fila++;

 const encOk = ["IPS", "Fecha actualización", "Insumos modificados", "Dosis consumidas", "Dosis repuestas", ""];
```

```
  hoja.getRange(fila, 1, 1, encOk.length).setValues([encOk])
    .setBackground(COLOR.VERDE_TEXTO).setFontColor(COLOR.BLANCO)
    .setFontWeight("bold").setFontSize(10);
  hoja.setRowHeight(fila, 20);
  fila++;

  correctas.forEach((ips, idx) => {
    hoja.getRange(fila, 1, 1, 6).setValues([[
      ips.nombre, ips.fecha_curr, `${ips.n_changed}/${ips.total_insumos}`,
      ips.consumidas > 0 ? `-${ips.consumidas}` : "—",
      ips.repuestas > 0 ? `+${ips.repuestas}` : "—", ""
    ]]).setBackground(idx % 2 === 0 ? COLOR.VERDE_FONDO : "#f0faf6")
      .setFontColor(COLOR.VERDE_TEXTO).setFontSize(10);
    hoja.setRowHeight(fila, 19);
    fila++;
  });

  // Formato columnas
  [180, 130, 130, 120, 120, 260].forEach((w, i) => hoja.setColumnWidth(i + 1, w));

  Logger.log("Hoja de alertas generada.");
}

//
═══════════════════════════════════════════════════════════════════════
═══
// 6. NOTIFICACIÓN POR CORREO ELECTRÓNICO
//
═══════════════════════════════════════════════════════════════════════
═══
```

```
function enviarEmailAlerta(analisis, snapAct, snapPrev) {
  const problemas = analisis.filter(i =>
["solo_fecha","sin_datos","no_actualizo"].includes(i.estado));
  const ok = analisis.filter(i => i.estado === "ok");
  const totalConsumidas = analisis.reduce((s, i) => s + i.consumidas, 0);
  const periodoStr = snapPrev
   ? `${snapPrev.fecha_humana} → ${snapAct.fecha_humana}`
   : snapAct.fecha_humana;

  const ss = SpreadsheetApp.openById(SHEET_ID);
  const urlSheet = ss.getUrl();
  const hoy = Utilities.formatDate(new Date(), "America/Bogota", "dd/MM/yyyy");

  const filasCriticas = problemas.map(ips => {
   const labels = { solo_fecha:"⚠️ Solo fecha", sin_datos:"⛔ Sin datos", no_actualizo:"🔴
No actualizó" };
   return `<tr style="background:${ips.estado!=='solo_fecha'?'#fcebeb':'#faeeda'}">
    <td style="padding:8px 12px;font-weight:600">${ips.nombre}</td>
    <td style="padding:8px 12px">${labels[ips.estado]}</td>
    <td style="padding:8px 12px">${ips.fecha_curr || "Sin fecha"}</td>
    <td style="padding:8px 12px">${ips.n_changed}/${ips.total_insumos} insumos</td>
   </tr>`;
  }).join("");

  const filasOk = ok.map(ips =>
   `<tr><td style="padding:6px 12px;color:#085041">✅ ${ips.nombre}</td>
    <td style="padding:6px 12px;color:#085041">${ips.fecha_curr}</td>
    <td style="padding:6px 12px;color:#085041">${ips.n_changed} insumos
modificados</td></tr>`
  ).join("");
```

```
const html = `
<div style="font-family:-apple-system,BlinkMacSystemFont,'Segoe UI',Arial,sans-serif;max-width:700px;margin:0 auto">
  <div style="background:#1a1a18;color:#fff;padding:20px 28px;border-radius:10px 10px 0 0">
    <h1 style="margin:0;font-size:20px;font-weight:600">📊 Informe Semanal de Inventario</h1>
    <p style="margin:6px 0 0;font-size:13px;color:#ccc">PAI Risaralda 2026 · Secretaría de Salud de Pereira</p>
  </div>

  <div style="background:#f5f5f0;padding:20px 28px;border-left:4px solid #185fa5">
    <p style="margin:0;font-size:13px;color:#555">
      <strong>Período:</strong> ${periodoStr}<br>
      <strong>Generado:</strong> ${hoy}
    </p>
  </div>

  ${problemas.length > 0 ? `
  <div style="background:#fcebeb;padding:16px 28px;border-left:4px solid #e24b4a;margin-top:4px">
    <h2 style="margin:0 0 4px;font-size:15px;color:#791f1f">⚠️ ${problemas.length} IPS con problemas de actualización</h2>
    <p style="margin:0;font-size:12px;color:#a32d2d">Requieren atención inmediata</p>
  </div>
  <table style="width:100%;border-collapse:collapse;margin-top:4px">
    <tr style="background:#791f1f;color:#fff">
      <th style="padding:8px 12px;text-align:left">IPS</th>
      <th style="padding:8px 12px;text-align:left">Estado</th>
      <th style="padding:8px 12px;text-align:left">Fecha registrada</th>
```

<th style="padding:8px 12px;text-align:left">Datos</th>

</tr>

${filasCriticas}

</table>` : `

✅ Todas las IPS actualizaron correctamente esta semana

`}

Resumen general

${analisis.length}

Total IPS

${ok.length}

Actualizaron

${problemas.length}

Con problemas

${totalConsumidas.toLocaleString("es-CO")}

```
      <div style="font-size:11px;color:#ba7517">Dosis consumidas</div>
    </div>
   </div>
  </div>

  <div style="padding:20px 28px">
   <a href="${urlSheet}" style="display:inline-block;background:#0f6e56;color:#fff;padding:12px 24px;border-radius:8px;text-decoration:none;font-size:13px;font-weight:600">
    📊 Ver informe completo en Google Sheets →
   </a>
  </div>

  <div style="padding:16px 28px;background:#f0f0e8;border-radius:0 0 10px 10px">
   <p style="margin:0;font-size:11px;color:#888">
    Este informe fue generado automáticamente por el Sistema de Alertas PAI Risaralda.<br>
    No responder a este correo. Para soporte, contacta al equipo de la Secretaría de Salud de Pereira.
   </p>
  </div>
 </div>`;

 try {
  GmailApp.sendEmail(
   CONFIG.EMAIL_ADMIN,
   `[PAI Risaralda] Informe semanal ${hoy} — ${problemas.length > 0 ? problemas.length + " IPS con problemas" : "Todo correcto ✅"}`,
   "Ver HTML adjunto",
   { htmlBody: html, name: "Sistema PAI Risaralda" }
  );
```

```
    Logger.log("Email enviado a " + CONFIG.EMAIL_ADMIN);
  } catch (e) {
    Logger.log("Error al enviar email: " + e.toString());
  }
}

//
═══════════════════════════════════════════════════════════════════
═══════
// 7. MENÚ PERSONALIZADO EN GOOGLE SHEETS
//
═══════════════════════════════════════════════════════════════════
═══════
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu("🚒 PAI Risaralda")
    .addItem("📸 Tomar snapshot ahora", "tomarSnapshot")
    .addSeparator()
    .addItem("📊 Generar informe semanal", "generarInformeCompleto")
    .addItem("📧 Enviar informe por correo", "enviarInformeManual")
    .addSeparator()
    .addItem("🗓️ Ver historial de snapshots", "verHistorial")
    .addItem("⚙️ Instalar triggers automáticos", "instalarTriggers")
    .addToUi();
}

function enviarInformeManual() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const historial = obtenerHistorial(ss);
  if (historial.length < 2) {
```

```
  SpreadsheetApp.getUi().alert("Necesitas al menos 2 snapshots para generar el comparativo.");
  return;
 }
 const analisis = analizarCambios(historial[0], historial[1]);
 enviarEmailAlerta(analisis, historial[0], historial[1]);
 SpreadsheetApp.getUi().alert("📧 Correo enviado a " + CONFIG.EMAIL_ADMIN);
}

function verHistorial() {
 const ss = SpreadsheetApp.openById(SHEET_ID);
 const historial = obtenerHistorial(ss);
 if (historial.length === 0) {
  SpreadsheetApp.getUi().alert("No hay snapshots registrados todavía.");
  return;
 }
 const lista = historial.map((s, i) =>
  `${i + 1}. ${s.fecha_humana} — ${Object.keys(s.datos).length} IPS`
 ).join("\n");
 SpreadsheetApp.getUi().alert("🗓️ Historial de snapshots:\n\n" + lista);
}

//
═══════════════════════════════════════════════════════════════
═══
// 8. FUNCIÓN DE PRUEBA RÁPIDA (para verificar que funciona)
//
═══════════════════════════════════════════════════════════════
═══
function prueba_TomarYGenerar() {
 // Toma snapshot actual, lo duplica como "anterior" y genera el informe
```

```
  // Útil para probar sin tener que esperar una semana
  const ss = SpreadsheetApp.openById(SHEET_ID);
  Logger.log("Tomando snapshot de prueba...");
  const snap1 = tomarSnapshot();

  // Simular una pequeña diferencia en el snap "anterior" para que el reporte sea interesante
  const snap2 = JSON.parse(JSON.stringify(snap1));
  snap2.ts = new Date(Date.now() - 7 * 24 * 60 * 60 * 1000).toISOString();
  snap2.fecha_humana = "Semana anterior (simulada)";
  // Guardar snap anterior manualmente
  guardarSnapshot(ss, snap2);

  Logger.log("Generando informe...");
  generarInformeCompleto();
  Logger.log("¡Prueba completada!");
}
```