```
/*****************************************
 * 0) Utilidades de normalización de textos
 *****************************************/
function _norm(s) { return String(s || '').trim().toUpperCase(); }
function _idx(headers, name) {
  const target = _norm(name);
  return headers.findIndex(h => _norm(h) === target); // -1 si no existe
}

function _fmtPct(value, total) {
  if (!total) return '0%';
  // 1 decimal (ej. 37.5%)
  return (Math.round((value * 1000) / total) / 10) + '%';
}
/*************************************
 * 1) Procesa respuestas del formulario
 *************************************/
function procesarRespuestas() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const FORM_SHEET_NAME = 'Respuestas de formulario 1';
  const RESUMEN_SHEET_NAME = 'Resumen';

  const formSheet = ss.getSheetByName(FORM_SHEET_NAME);
  if (!formSheet) throw new Error(`No existe la hoja
"${FORM_SHEET_NAME}".`);

  let resumenSheet = ss.getSheetByName(RESUMEN_SHEET_NAME);
  if (!resumenSheet) resumenSheet = ss.insertSheet(RESUMEN_SHEET_NAME);

  // Encabezados estándar para "Resumen"
  const headers = [
    "Nombre", "Email", "Teléfono",
    "USTED Tierra", "USTED Fuego", "USTED Agua", "USTED Aire",
    "OTRO Tierra", "OTRO Fuego", "OTRO Agua", "OTRO Aire",
    "RESULTADO YO", "RESULTADO OTRO"
  ];

  // Limpiar y escribir encabezados
  resumenSheet.clearContents();
```

```
  resumenSheet.getRange(1, 1, 1, headers.length).setValues([headers]);

  const data = formSheet.getDataRange().getValues();
  if (!data || data.length <= 1) return; // sin respuestas

  const rows = data.slice(1);
  const out = [];

  rows.forEach(row => {
    const email = row[1] || "";
    const nombre = row[2] || "";
    const telefono = row[3] || "";

    let yo = { Tierra: 0, Fuego: 0, Agua: 0, Aire: 0 };
    let otro = { Tierra: 0, Fuego: 0, Agua: 0, Aire: 0 };

    // Contabiliza USTED / OTRA PERSONA por bloques de 8
    // Ajusta los índices si tu formulario cambia de orden/longitud.
    const countRange = (start, end, key) => {
      for (let i = start; i <= end && i < row.length; i++) {
        const v = _norm(row[i]);
        if (v === 'USTED') yo[key]++;
        else if (v === 'OTRA PERSONA') otro[key]++;
      }
    };

    // Bloques esperados (8 ítems cada uno)
    countRange(4, 11, 'Tierra');
    countRange(12, 19, 'Fuego');
    countRange(20, 27, 'Agua');
    countRange(28, 35, 'Aire');

    const maxYo = Math.max(...Object.values(yo));
    const maxOtro = Math.max(...Object.values(otro));
    const ganadorYo = Object.keys(yo).filter(k => yo[k] ===
maxYo).join("/");
    const ganadorOtro = Object.keys(otro).filter(k => otro[k] ===
maxOtro).join("/");
```

```
    out.push([
      email, nombre, telefono,
      yo.Tierra, yo.Fuego, yo.Agua, yo.Aire,
      otro.Tierra, otro.Fuego, otro.Agua, otro.Aire,
      ganadorYo, ganadorOtro
    ]);
  });

  if (out.length) {
    resumenSheet.getRange(2, 1, out.length, headers.length).setValues(out);
  }
}

/**********************
 * 2) SETUP AUTOMÁTICO
 **********************/
function setupAutomation() {
  borrarTriggersPorHandler_('pipelineOnSubmit');
  borrarTriggersPorHandler_('onEditHandler');

  const ssId = SpreadsheetApp.getActiveSpreadsheet().getId();

  ScriptApp.newTrigger('pipelineOnSubmit')
    .forSpreadsheet(ssId)
    .onFormSubmit()
    .create();

  ScriptApp.newTrigger('onEditHandler')
    .forSpreadsheet(ssId)
    .onEdit()
    .create();

  SpreadsheetApp.getUi().alert('✅ Automatización instalada:\n• onFormSubmit → pipelineOnSubmit\n• onEdit → onEditHandler');
}

function borrarTriggersPorHandler_(handler) {
  ScriptApp.getProjectTriggers()
    .filter(t => t.getHandlerFunction() === handler)
```

```
    .forEach(t => ScriptApp.deleteTrigger(t));
}

/*******************************************
 * 3) PIPELINE: se ejecuta al enviar formulario
 *******************************************/
function pipelineOnSubmit() {
  procesarRespuestas();
  ensureResumenPrepared_();
  generarPastelYo();
}

/*******************************************
 * 4) onEdit INSTALABLE: abre modal al marcar "Ver"
 *******************************************/
function onEditHandler(e) {
  try {
    if (!e || !e.range) return;
    const sh = e.range.getSheet();
    if (!sh || sh.getName() !== 'Resumen') return;

    // Asegura existencia/estado de la columna "Ver"
    ensureResumenPrepared_();

    const editedRow = e.range.getRow();
    const editedCol = e.range.getColumn();
    if (editedRow === 1) return; // Encabezado no

    const lastCol = sh.getLastColumn();
    const headers = sh.getRange(1, 1, 1, lastCol).getValues()[0];
    const colVer = _idx(headers, 'VER') + 1; // 0 si no existe
    if (!colVer || editedCol !== colVer) return;

    const value = e.range.getValue();
    if (value === true) {
      verGraficosFila(editedRow);
      // Desmarcar para permitir re-abrir en el futuro
      e.range.setValue(false);
    }
```

```
  } catch (err) {
    console.error(err);
  }
}

/**********************************************************
 * 5) Helper "self-healing": asegura la columna Ver/checkbox
 **********************************************************/
function ensureResumenPrepared_() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sh = ss.getSheetByName('Resumen');
  if (!sh) return;

  const lastRow = sh.getLastRow();
  const lastCol = sh.getLastColumn();
  if (lastRow < 1) return;

  const headers = sh.getRange(1, 1, 1, lastCol).getValues()[0];
  let verCol = _idx(headers, 'VER') + 1;

  // Crear encabezado "Ver" si no existe
  if (!verCol) {
    verCol = lastCol + 1;
    sh.getRange(1, verCol).setValue('Ver');
  }

  // Poner/actualizar checkboxes desde fila 2 hasta la última con datos reales (col 1)
  const dataLastRow = sh.getLastRow();
  const n = Math.max(0, dataLastRow - 1);
  if (n > 0) {
    const rng = sh.getRange(2, verCol, n, 1);
    rng.insertCheckboxes();
    // Si quieres dejar todo en false por defecto, descomenta:
    // rng.setValue(false);
  }
}

/**************************************
```

```
 * 6) UI opcional (menú) para gestionar setup
 ****************************************/
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('Resumen')
    .addItem('Instalar automatización', 'setupAutomation')
    .addSeparator()
    .addItem('Ver gráfico de fila seleccionada',
'verGraficosFilaSeleccionada')
    .addItem('Exportar PDF (todos)', 'exportarPDFResumen') //
    .addToUi();
}

/************************************************
 * 7) Ver gráficos (USTED vs OTRO) para una FILA
 ************************************************/
function verGraficosFilaSeleccionada() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sh = ss.getSheetByName('Resumen');
  if (!sh) { SpreadsheetApp.getUi().alert('No existe la hoja "Resumen".');
return; }
  const r = sh.getActiveRange();
  if (!r) { SpreadsheetApp.getUi().alert('Selecciona una celda de la
fila.'); return; }
  if (r.getRow() === 1) { SpreadsheetApp.getUi().alert('Selecciona una fila
de datos.'); return; }
  verGraficosFila(r.getRow());
}

function verGraficosFila(row) {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sh = ss.getSheetByName('Resumen');
  if (!sh) throw new Error('No existe la hoja "Resumen".');

  const lastCol = sh.getLastColumn();
  const headers = sh.getRange(1, 1, 1, lastCol).getValues()[0];

  const colUstedTierra = _idx(headers, 'USTED TIERRA') + 1;
  const colUstedFuego = _idx(headers, 'USTED FUEGO') + 1;
```

```
  const colUstedAgua = _idx(headers, 'USTED AGUA') + 1;
  const colUstedAire = _idx(headers, 'USTED AIRE') + 1;

  const colOtroTierra = _idx(headers, 'OTRO TIERRA') + 1;
  const colOtroFuego = _idx(headers, 'OTRO FUEGO') + 1;
  const colOtroAgua = _idx(headers, 'OTRO AGUA') + 1;
  const colOtroAire = _idx(headers, 'OTRO AIRE') + 1;

  if ([colUstedTierra, colUstedFuego, colUstedAgua, colUstedAire,
colOtroTierra, colOtroFuego, colOtroAgua, colOtroAire].some(c => c === 0))
{
    throw new Error('Faltan columnas: USTED/OTRO Tierra/Fuego/Agua/Aire.');
  }

  const usted = [
    ['Elemento', 'Cantidad'],
    ['Tierra', Number(sh.getRange(row, colUstedTierra).getValue()) || 0],
    ['Fuego', Number(sh.getRange(row, colUstedFuego).getValue()) || 0],
    ['Agua', Number(sh.getRange(row, colUstedAgua).getValue()) || 0],
    ['Aire', Number(sh.getRange(row, colUstedAire).getValue()) || 0],
  ];

  const otro = [
    ['Elemento', 'Cantidad'],
    ['Tierra', Number(sh.getRange(row, colOtroTierra).getValue()) || 0],
    ['Fuego', Number(sh.getRange(row, colOtroFuego).getValue()) || 0],
    ['Agua', Number(sh.getRange(row, colOtroAgua).getValue()) || 0],
    ['Aire', Number(sh.getRange(row, colOtroAire).getValue()) || 0],
  ];

  // Datos opcionales para el título
  const colEmail = _idx(headers, 'EMAIL') + 1;
  const colNombre = _idx(headers, 'NOMBRE') + 1;
  const email = colEmail ? String(sh.getRange(row, colEmail).getValue() ||
'') : '';
  const nombre = colNombre ? String(sh.getRange(row, colNombre).getValue()
|| '') : '';

  const html = buildDialogHtml_(usted, otro, nombre, email, row);
```

```
  const output =
HtmlService.createHtmlOutput(html).setWidth(720).setHeight(480);
  SpreadsheetApp.getUi().showModalDialog(output, 'Distribución de
estilos');
}

/**********************************
 * 8) HTML del diálogo de gráficos
 **********************************/
function buildDialogHtml_(ustedData, otroData, nombre, email, row) {
  const encode = (obj) => JSON.stringify(obj);
  const title = [nombre, email].filter(Boolean).join(' • ') || ('Fila ' +
row);

  return `
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script src="https://www.gstatic.com/charts/loader.js"></script>
  <style>
    body { font-family: Arial, sans-serif; margin: 16px; }
    h2 { margin: 0 0 12px 0; font-size: 16px; color: #444; }
    .wrap { display: grid; grid-template-columns: 1fr 1fr; gap: 16px; }
    .card { border: 1px solid #e5e5e5; border-radius: 8px; padding: 12px; }
    #ustedChart, #otroChart { width: 100%; height: 320px; }
    .header { margin-bottom: 6px; font-weight: bold; color: #222; }
    .sub { color: #666; font-size: 12px; margin-bottom: 12px; }
  </style>
</head>
<body>
  <div class="header">${title}</div>
  <div class="sub">Comparativo de distribución por estilos</div>
  <div class="wrap">
    <div class="card">
      <h2>USTED</h2>
      <div id="ustedChart"></div>
    </div>
    <div class="card">
```

```
      <h2>OTRO</h2>
      <div id="otroChart"></div>
    </div>
  </div>

  <script>
    const ustedData = ${encode(ustedData)};
    const otroData  = ${encode(otroData)};

    // Paleta fija por etiqueta
    // Ajusta los hex si deseas otra gama.
    const PALETTE = {
      Tierra: '#8BC68B',  // verde
      Fuego:  '#F49AA0',  // rosa/salmón
      Agua:   '#8EC5F0',  // celeste
      Aire:   '#FFEE8C',  // naranja
    };

    // Dado un DataTable, devuelve los colores en el orden de las etiquetas
de la 1ª columna
    function colorsFor(dt) {
      const n = dt.getNumberOfRows();
      const colors = [];
      for (let i = 0; i < n; i++) {
        const label = dt.getValue(i, 0); // nombre del elemento
        colors.push(PALETTE[label] || '#cccccc'); // fallback gris
      }
      return colors;
    }

    google.charts.load('current', {'packages':['corechart']});
    google.charts.setOnLoadCallback(draw);

    function draw() {
      const usted = google.visualization.arrayToDataTable(ustedData);
      const otro  = google.visualization.arrayToDataTable(otroData);

      const optsBase = {
        titleTextStyle: { fontSize: 14 },
```

```
        legend: { position: 'right' },
        chartArea: { left: 10, top: 10, width: '80%', height: '85%' },
        pieSliceText: 'value' // no cambio funcionalidad
        // pieHole: 0.35 // donut (opcional)
      };

      const optsUsted = Object.assign({ title: 'USTED', colors:
colorsFor(usted) }, optsBase);
      const optsOtro  = Object.assign({ title: 'OTRO',  colors:
colorsFor(otro)  }, optsBase);

      new
google.visualization.PieChart(document.getElementById('ustedChart')).draw(u
sted, optsUsted);
      new
google.visualization.PieChart(document.getElementById('otroChart')).draw(ot
ro,  optsOtro);
    }

    // Redibuja si cambia el tamaño de la ventana
    window.addEventListener('resize', draw);
  </script>
</body>
</html>`;
}


/**********************************************
 * 9) Genera hoja "Totales" con gráfico pastel
 **********************************************/
function generarPastelYo() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const RESUMEN = 'Resumen';
  const TOTALES = 'Totales';

  const resumen = ss.getSheetByName(RESUMEN);
  if (!resumen) throw new Error(`No existe la hoja "${RESUMEN}". Ejecuta
primero procesarRespuestas().`);
```

```
  const lastRow = resumen.getLastRow();
  const lastCol = resumen.getLastColumn();
  if (lastRow < 2) throw new Error('No hay datos en "Resumen".');

  const headers = resumen.getRange(1, 1, 1, lastCol).getValues()[0];
  const colResultadoYo = _idx(headers, 'RESULTADO YO') + 1;
  if (colResultadoYo === 0) throw new Error('No se encontró la columna
"RESULTADO YO".');

  const valores = resumen.getRange(2, colResultadoYo, lastRow - 1,
1).getValues();

  const conteo = { Tierra: 0, Fuego: 0, Agua: 0, Aire: 0 };
  const mapKey = (s) => {
    const t = _norm(s);
    if (t === 'TIERRA') return 'Tierra';
    if (t === 'FUEGO') return 'Fuego';
    if (t === 'AGUA') return 'Agua';
    if (t === 'AIRE') return 'Aire';
    return null;
  };

  const desconocidos = [];

  valores.forEach(([valor]) => {
    if (!valor) return;
    String(valor).split('/').forEach(part => {
      const key = mapKey(part);
      if (key) conteo[key] += 1;
      else desconocidos.push(String(part).trim());
    });
  });

  // --- TOTAL para porcentajes ---
  const total = conteo.Tierra + conteo.Fuego + conteo.Agua + conteo.Aire;
  const fmtPct = (v) => total ? (Math.round(v * 1000 / total) / 10) + '%' :
'0%'; // 1 decimal

  // Crear/obtener hoja "Totales"
```

```
  let totales = ss.getSheetByName(TOTALES);
  if (!totales) totales = ss.insertSheet(TOTALES);

  // (1) Quitar charts existentes ANTES de limpiar celdas
  totales.getCharts().forEach(c => totales.removeChart(c));

  // (2) Limpiar celdas
  totales.clear();

  // Etiquetas enriquecidas: "Tierra — 7 (38.9%)"
  const tabla = [
    ['Elemento', 'Cantidad de personas'],
    [`Tierra — ${conteo.Tierra} (${fmtPct(conteo.Tierra)})`,
conteo.Tierra],
    [`Fuego — ${conteo.Fuego} (${fmtPct(conteo.Fuego)})`, conteo.Fuego],
    [`Agua — ${conteo.Agua} (${fmtPct(conteo.Agua)})`, conteo.Agua],
    [`Aire — ${conteo.Aire} (${fmtPct(conteo.Aire)})`, conteo.Aire],
  ];
  totales.getRange(1, 1, tabla.length, tabla[0].length).setValues(tabla);

  if (desconocidos.length) {
    totales.getRange(1, 4, 1, 1).setValue('Valores no reconocidos');
    totales.getRange(2, 4, desconocidos.length,
1).setValues(desconocidos.map(x => [x]));
  }

  // ===== Colores predeterminados (Tierra, Fuego, Agua, Aire) =====
  const COLORS = {
    Tierra: '#8BC68B',  // verde
    Fuego:  '#F49AA0',  // rosa/salmón
    Agua:   '#8EC5F0',  // celeste
    Aire:   '#FFEE8C',  // naranja
  };

  const pieColors = [COLORS.Tierra, COLORS.Fuego, COLORS.Agua,
COLORS.Aire];

  // Insertar gráfico de pastel con porcentaje+valor y colores fijos
  const chart = totales.newChart()
```

```
    .setChartType(Charts.ChartType.PIE)
    .addRange(totales.getRange('A2:B5'))
    .setOption('title', 'Distribución de tipos de comunicación - EDT')
    .setOption('titleTextStyle', { fontSize: 17, bold: true, color:
'#333333' })
    .setOption('pieSliceText', 'percentage')
    .setOption('legend', { position: 'right' })
    .setOption('colors', pieColors)               // ← colores
predeterminados
    .setPosition(1, 4, 0, 0)
    .build();

  totales.insertChart(chart);
}



/*******************************************************
 * 10) Exporta PDF con pares de pasteles (USTED vs OTRO)
 *     Una página por persona: Nombre + 2 gráficos
 *******************************************************/
function exportarPDFResumen() {
  const ssOrigen = SpreadsheetApp.getActiveSpreadsheet();
  const SH_RESUMEN = 'Resumen';
  const resumen = ssOrigen.getSheetByName(SH_RESUMEN);
  if (!resumen) throw new Error(`No existe la hoja "${SH_RESUMEN}". Ejecuta
primero procesarRespuestas().`);

  const lastRow = resumen.getLastRow();
  const lastCol = resumen.getLastColumn();
  if (lastRow < 2) throw new Error('No hay datos en "Resumen".');

  const headers = resumen.getRange(1, 1, 1, lastCol).getValues()[0];

  // --- Columnas por encabezado ---
  const colNombre = _idx(headers, 'NOMBRE') + 1;
  const colUTierra = _idx(headers, 'USTED TIERRA') + 1;
  const colUFuego = _idx(headers, 'USTED FUEGO') + 1;
  const colUAgua = _idx(headers, 'USTED AGUA') + 1;
```

```
  const colUAire = _idx(headers, 'USTED AIRE') + 1;
  const colOTierra = _idx(headers, 'OTRO TIERRA') + 1;
  const colOFuego = _idx(headers, 'OTRO FUEGO') + 1;
  const colOAgua = _idx(headers, 'OTRO AGUA') + 1;
  const colOAire = _idx(headers, 'OTRO AIRE') + 1;

  const faltantes = [
    ['USTED TIERRA', colUTierra], ['USTED FUEGO', colUFuego], ['USTED
AGUA', colUAgua], ['USTED AIRE', colUAire],
    ['OTRO TIERRA', colOTierra], ['OTRO FUEGO', colOFuego], ['OTRO AGUA',
colOAgua], ['OTRO AIRE', colOAire]
  ].filter(([_, c]) => c === 0);
  if (faltantes.length) throw new Error('Faltan columnas en "Resumen": ' +
faltantes.map(f => f[0]).join(', '));

  // === Spreadsheet temporal ===
  const ssTemp = SpreadsheetApp.create('Resultados Temporales');

  // ===== DISEÑO (vertical) =====
  const PAGE_SIZE = 'A3';
  const PORTRAIT = true;
  const ROW_PX = 18;          // altura de fila
  const MERGE_LAST_COL = 'K';
  const UNIFORM_COL_W = 90;

  // Tamaño del gráfico (PNG). Mantén moderado para PDF limpio
  const IMG_W = 560; // px
  const IMG_H = 360; // px

  // Layout “lado a lado”
  const CHART_COL = 2;                // columna B (gráfico)
  const TABLE_COL = 9;                // columna I (tabla)
  const BLOCK1_START_ROW = 8;         // fila de inicio del bloque USTED
  const GAP_AFTER_BLOCK = 6;         // filas de aire entre bloques

  const COLORES = ['#8BC68B', '#F49AA0', '#8EC5F0', '#FFEE8C'];

  // ==== Helpers locales (incluye interpretar y ponerTextoCentrado) ====
```

```
  function colLetraANum(col) { let n = 0; for (let i = 0; i < col.length;
i++) n = n * 26 + (col.charCodeAt(i) - 64); return n; }
  function prepararHoja(hoja) {
    hoja.setColumnWidth(1, 130); // A
    hoja.setColumnWidth(2, 90);  // B
    const fromC = 3, toK = colLetraANum(MERGE_LAST_COL);
    hoja.setColumnWidths(fromC, toK - fromC + 1, UNIFORM_COL_W);
    hoja.setRowHeights(1, 1000, ROW_PX);
  }
  function ponerTextoCentrado(hoja, row, texto, isBold, fontSize,
fontFamily) {
    const rango = hoja.getRange(`A${row}:${MERGE_LAST_COL}${row}`).merge();
    rango.setValue(texto)
      .setHorizontalAlignment('center')
      .setVerticalAlignment('middle')
      .setWrap(true);
    if (isBold) rango.setFontWeight('bold');
    if (fontSize) rango.setFontSize(fontSize);
    if (fontFamily) rango.setFontFamily(fontFamily);
  }
  function pctES(v, total) {
    const p = total > 0 ? (v / total) * 100 : 0;
    const s = (Math.round(p * 10) / 10).toString();
    const txt = s.indexOf('.') >= 0 ? s.replace('.', ',') : s;
    return txt.replace(',0', '') + '%';
  }
  function scaleToPixelCap(w, h, maxPx) {
    const px = w * h; if (px <= maxPx) return { w, h };
    const s = Math.sqrt(maxPx / px); return { w: Math.floor(w * s), h:
Math.floor(h * s) };
  }
  function crearPieBlob(titulo, t, f, a, r, w, h) {
    const { w: W, h: H } = scaleToPixelCap(w, h, 950000);
    const labels = ['Tierra', 'Fuego', 'Agua', 'Aire'];
    const vals = [t, f, a, r].map(x => Number(x) || 0);
    const total = Math.max(1, vals.reduce((x, y) => x + y, 0));
    const legendLabels = vals.map((v, i) => `${labels[i]} — ${v}
(${pctES(v, total)})`);
    const dt = Charts.newDataTable()
```

```
      .addColumn(Charts.ColumnType.STRING, 'Elemento')
      .addColumn(Charts.ColumnType.NUMBER, 'Cantidad')
      .addRow([legendLabels[0], vals[0]])
      .addRow([legendLabels[1], vals[1]])
      .addRow([legendLabels[2], vals[2]])
      .addRow([legendLabels[3], vals[3]])
      .build();
    const pie = Charts.newPieChart()
      .setDataTable(dt).setDimensions(W, H).setColors(COLORES)
      .setLegendPosition(Charts.Position.RIGHT).setTitle(titulo || '')
      .setOption('pieSliceText', 'value-and-percentage')
      .setOption('pieSliceTextStyle', { fontSize: 14, bold: true, color:
'#000' })
      .setOption('pieSliceBorderColor', '#fff')
      .setOption('titleTextStyle', { fontSize: 14, bold: true, color:
'#333' })
      .setOption('legend', { position: 'right', alignment: 'center',
textStyle: { fontSize: 12, color: '#333' } })
      .setOption('chartArea', { left: 10, top: 26, width: '72%', height:
'78%' }).build();
    return { blob: pie.getAs('image/png'), W, H };
  }
  function insertarImagenEn(hoja, blob, col, row, w, h) {
    const img = hoja.insertImage(blob, col, row);
    img.setWidth(w).setHeight(h);
    const filasOcupadas = Math.ceil(h / ROW_PX);
    return row + filasOcupadas - 1; // última fila ocupada
  }
  function escribirTablaResultados(hoja, startRow, startCol, t, f, a, r) {
    const total = t + f + a + r || 0;
    const data = [
      ['Elemento', 'Cantidad', '%'],
      ['Tierra', t, pctES(t, total)],
      ['Fuego', f, pctES(f, total)],
      ['Agua', a, pctES(a, total)],
      ['Aire', r, pctES(r, total)],
    ];
    const rng = hoja.getRange(startRow, startCol, data.length,
data[0].length);
```

```
    rng.setValues(data);
    hoja.getRange(startRow, startCol, 1,
3).setFontWeight('bold').setBackground('#f5f5f5');
    hoja.getRange(startRow + 1, startCol + 1, 4,
1).setHorizontalAlignment('center');
    hoja.getRange(startRow + 1, startCol + 2, 4,
1).setHorizontalAlignment('center');
    hoja.getRange(startRow, startCol, data.length,
data[0].length).setBorder(true, true, true, true, true, true);

    // >>> margen derecho: columnas vacías con ancho específico
    const tableCols = data[0].length; // 3
    return startRow + data.length - 1;
  }

  function nombreHojaSeguro(base) {
    const s = String(base || 'Resultado').replace(/[\\/:*?\[\]]/g,
'_').slice(0, 90);
    return s || 'Resultado';
  }
  function joinHumano(arr) {
    if (arr.length === 1) return arr[0];
    if (arr.length === 2) return `${arr[0]} y ${arr[1]}`;
    return `${arr.slice(0, -1).join(', ')} y ${arr[arr.length - 1]}`;
  }
  function interpretar(t, f, a, r) {
    const defs = {
      'Tierra': 'El ritmo más lento de Tierra se debe a que necesita mucha
información formal (“aterrizada”) para estar seguro. Requieren de
información estructurada, de datos y precisión, y conocer todos los
posibles riesgos antes de tomar una decisión. No les gustan las
improvisaciones ni las prisas.',
      'Fuego': 'El estilo de comunicación Fuego es rápido en sus decisiones
porque valora mucho su tiempo. Le aburren los detalles, lo que quiere es un
“titular” breve y acertado de las ventajas del producto que se le
propone.',
      'Agua': 'En el caso de Agua, su proceso de decisión pasa primero por
construir la relación. El conflicto con otros le paraliza mientras que los
ambientes fluidos y amigables sacan lo mejor de él/ella.',
```

```
      'Aire': 'Se entusiasma con las ideas que le hacen soñar, con las imágenes vívidas que le permiten visualizar los proyectos y productos. No necesita detalles ni informaciones precisas, lo que le motiva es seguir creando a partir de las primeras impresiones que le llegan.'
    };
    const pares = [
      { k: 'Tierra', v: Number(t) || 0 },
      { k: 'Fuego', v: Number(f) || 0 },
      { k: 'Agua', v: Number(a) || 0 },
      { k: 'Aire', v: Number(r) || 0 },
    ].sort((x, y) => y.v - x.v);
    const max = pares[0].v;
    const winners = pares.filter(p => p.v === max).map(p => p.k);
    if (winners.length === 1) {
      const tipo = winners[0];
      return { tipo, texto: `${tipo}: ${defs[tipo]}` };
    } else {
      const tipo = `Combinación: ${joinHumano(winners)}`;
      const texto = `Integra rasgos de ${joinHumano(winners.map(k => k.toLowerCase()))}.`;
      return { tipo, texto };
    }
  }

  // ===== Recorrer filas de "Resumen" =====
  for (let r = 2; r <= lastRow; r++) {
    const nombre = colNombre ? String(resumen.getRange(r, colNombre).getValue() || '') : '';

    const uT = Number(resumen.getRange(r, colUTierra).getValue()) || 0;
    const uF = Number(resumen.getRange(r, colUFuego).getValue()) || 0;
    const uG = Number(resumen.getRange(r, colUAgua).getValue()) || 0;
    const uA = Number(resumen.getRange(r, colUAire).getValue()) || 0;

    const oT = Number(resumen.getRange(r, colOTierra).getValue()) || 0;
    const oF = Number(resumen.getRange(r, colOFuego).getValue()) || 0;
    const oG = Number(resumen.getRange(r, colOAgua).getValue()) || 0;
    const oA = Number(resumen.getRange(r, colOAire).getValue()) || 0;
```

```
    const sumaU = uT + uF + uG + uA;
    const sumaO = oT + oF + oG + oA;
    if (sumaU === 0 && sumaO === 0) continue;

    const hoja = ssTemp.insertSheet(`Resultado_${r -
1}_${nombreHojaSeguro(nombre)}`);
    prepararHoja(hoja);
    const TABLE_TOP_MARGIN_ROWS = 5;

    // Títulos
    ponerTextoCentrado(hoja, 1, 'Resultados Test de Comunicación EDT',
true, 18);
    ponerTextoCentrado(hoja, 2, `Nombre: ${nombre || '(sin nombre)'}`,
true, 16, 'Garamond');

    // === BLOQUE 1: USTED (gráfico izq + tabla der) ===
    const g1 = crearPieBlob('USTED', uT, uF, uG, uA, IMG_W, IMG_H);
    const lastRowChart1 = insertarImagenEn(hoja, g1.blob, CHART_COL,
BLOCK1_START_ROW, IMG_W, IMG_H);
    const lastRowTable1 = escribirTablaResultados(
      hoja,
      BLOCK1_START_ROW + TABLE_TOP_MARGIN_ROWS,
      TABLE_COL,
      uT, uF, uG, uA
    );
    // Interpretación USTED centrada debajo de su fila
    const interU = interpretar(uT, uF, uG, uA);
    const inter1TitleRow = Math.max(lastRowChart1, lastRowTable1) + 1;
    ponerTextoCentrado(hoja, inter1TitleRow, 'Interpretación', true, 13);
    ponerTextoCentrado(hoja, inter1TitleRow + 1, `${interU.texto}`, false,
13);

    // Inicio del bloque 2 por debajo de la interpretación USTED
    const block2Start = inter1TitleRow + 1 + GAP_AFTER_BLOCK;

    // === BLOQUE 2: OTRA PERSONA (gráfico izq + tabla der) ===
    const g2 = crearPieBlob('OTRA PERSONA', oT, oF, oG, oA, IMG_W, IMG_H);
    const lastRowChart2 = insertarImagenEn(hoja, g2.blob, CHART_COL,
block2Start, IMG_W, IMG_H);
```

```
    const lastRowTable2 = escribirTablaResultados(
      hoja,
      block2Start + TABLE_TOP_MARGIN_ROWS,
      TABLE_COL,
      oT, oF, oG, oA
    );
    // Interpretación OTRA PERSONA centrada debajo de su fila
    const inter0 = interpretar(oT, oF, oG, oA);
    const inter2TitleRow = Math.max(lastRowChart2, lastRowTable2) + 1;
    ponerTextoCentrado(hoja, inter2TitleRow, 'Interpretación', true, 13);
    ponerTextoCentrado(hoja, inter2TitleRow + 1, `${inter0.texto}`, false,
13);
  }

  // Quitar hoja vacía inicial si existe
  const hojas = ssTemp.getSheets();
  if (hojas.length > 1) {
    const h0 = hojas[0].getName();
    if (h0 === 'Hoja 1' || h0 === 'Sheet1') ssTemp.deleteSheet(hojas[0]);
  }

  SpreadsheetApp.flush();

  // === Exportar a PDF ===
  const exportUrl = ssTemp.getUrl().replace(/edit$/, '') +
    'export?format=pdf&exportFormat=pdf' +
    `&size=${PAGE_SIZE}` + `&portrait=${PORTRAIT ? 'true' : 'false'}` +

'&fitw=true&sheetnames=false&printtitle=false&pagenumbers=false&gridlines=f
alse';

  const token = ScriptApp.getOAuthToken();
  const response = UrlFetchApp.fetch(exportUrl, { headers: {
'Authorization': `Bearer ${token}` } });
  const blob = response.getBlob().setName('Resultados Test de Comunicación
EDT.pdf');
  const archivoPDF = DriveApp.createFile(blob);

  // Enviar el temporal a la papelera
```

```
  DriveApp.getFileById(ssTemp.getId()).setTrashed(true);

  const url =
`https://drive.google.com/uc?export=download&id=${archivoPDF.getId()}`;
  const html = HtmlService.createHtmlOutput(
    `<p><b>PDF generado con éxito</b></p>
     <a href="${url}" target="_blank">Descargar PDF</a>`
  ).setWidth(300).setHeight(100);
  SpreadsheetApp.getUi().showModalDialog(html, 'Descarga disponible');
}
```