Manual Técnico – Portal de Solicitudes (Grupo Fiancar)

Autor: Rodrigo Ilundain
Fecha de generación: 28/10/2025 20:52:36

1. Descripción general

Sistema integral de gestión de adelantos, licencias y licencias por estudio con registro automático, control de topes, feriados, historial y notificación por correo.

2. Estructura del proyecto

• Code.gs → Lógica del servidor (Google Apps Script)
• SolicitudAdelantoApp.html → Interfaz web (portal)
• Hoja 'Usuarios' → Base de datos principal
• Hojas mensuales ('Octubre 2025', 'Licencia Octubre 2025', etc.) → Registro de movimientos

3. Flujo general de uso

1. El usuario accede al menú 'Solicitudes → Iniciar solicitud'.
2. Se abre el portal HTML.
3. Puede registrarse, iniciar sesión, solicitar adelantos o licencias.
4. El sistema crea registros automáticos en las hojas correspondientes y envía notificaciones por correo.

11. Lógica técnica del sistema (para desarrolladores)

El portal funciona como una WebApp de Google Apps Script (GAS):
• Frontend: HTML + CSS + JS embebidos.
• Backend: Code.gs con funciones expuestas vía google.script.run.
• Base de datos: Google Sheets.
• Notificaciones: MailApp.sendEmail() con plantillas HTML.

12. Lógica de implementación (Code.gs)

A continuación se detalla la estructura funcional del backend implementado en Google Apps Script, incluyendo las funciones principales y su relación lógica dentro del flujo de solicitudes.

12.1 Funciones utilitarias

```
function getSheetById_(ss, sheetId) {
  const sheets = ss.getSheets();
  for (const sh of sheets) if (sh.getSheetId() === sheetId) return sh;
  return null;
}

function headerMap_(hoja) {
```

```
  const headers = hoja.getRange(1, 1, 1, hoja.getLastColumn()).getValues()[0];
  const map = {};
  headers.forEach((h, i) => {
    const key = h.toString().trim().toUpperCase().replace(/\s+/g, '_');
    map[key] = i + 1;
  });
  return map;
}

function stampNow_() {
  return Utilities.formatDate(new Date(), "America/Montevideo", "dd/MM/yyyy HH:mm:ss");
}

function countDiasHabiles_(desde, hasta) {
  const feriados = new Set(["01-01", "05-01", "07-18", "08-25", "12-25"]);
  let count = 0;
  for (let d = new Date(desde); d <= hasta; d.setDate(d.getDate() + 1)) {
    const dia = d.getDay();
    const key = Utilities.formatDate(d, "America/Montevideo", "MM-dd");
    if (dia > 0 && dia < 6 && !feriados.has(key)) count++;
  }
  return count;
}
```

12.2 Gestión de usuarios

```
function verificarUsuario(ci) {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const hoja = ss.getSheetByName("Usuarios");
  const c = headerMap_(hoja);
  const data = hoja.getDataRange().getValues();
  const user = data.find(r => r[c.CI - 1] === ci);
  return user ? {existe:true, fila:data.indexOf(user)+1, nombre:user[c.NOMBRE-1]} : {existe:
}

function registrarNuevoUsuario(datos) {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const hoja = ss.getSheetByName("Usuarios");
  const c = headerMap_(hoja);
  const row = hoja.getLastRow() + 1;
  hoja.getRange(row, c.CI).setValue(datos.ci);
  hoja.getRange(row, c.NOMBRE).setValue(datos.nombre);
  hoja.getRange(row, c.CARGO).setValue(datos.cargo);
  registrarFechaTimestamp_(hoja, row, c);
}

function registrarFechaTimestamp_(hoja, fila, c) {
  if (c.FECHA_REGISTRO)
    hoja.getRange(fila, c.FECHA_REGISTRO).setValue(stampNow_());
}
```

12.3 Gestión de solicitudes

```
function enviarAdelanto(payload) {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const hoja = ss.getSheetByName(payload.hoja);
  const c = headerMap_(hoja);
  hoja.appendRow([
    stampNow_(), payload.ci, payload.nombre, payload.monto,
```

```
    payload.motivo, payload.estado, payload.gerente
  ]);
  enviarEmailNotificacion(payload);
}

function enviarLicencia(payload) {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const hoja = ss.getSheetByName(payload.hoja);
  const c = headerMap_(hoja);
  const diasHabiles = countDiasHabiles_(new Date(payload.desde), new Date(payload.hasta));
  hoja.appendRow([
    stampNow_(), payload.ci, payload.nombre, payload.desde,
    payload.hasta, diasHabiles, payload.tipo, payload.gerente
  ]);
  enviarEmailNotificacion(payload);
}
```

12.4 Envío de notificaciones

```
function enviarEmailNotificacion(datos) {
  const html = `
    <p>Hola ${datos.gerente},</p>
    <p>El colaborador <b>${datos.nombre}</b> ha registrado una nueva solicitud.</p>
    <p><b>Tipo:</b> ${datos.tipo}<br><b>Fecha:</b> ${stampNow_()}</p>
    <p><a href="https://docs.google.com/spreadsheets/d/ID/..." target="_blank">Abrir registr
  MailApp.sendEmail({
    to: datos.emailGerente,
    cc: datos.emailRRHH,
    subject: `Nueva solicitud de ${datos.tipo} – ${datos.nombre}`,
    htmlBody: html
  });
}
```

Manual Técnico – Portal de Solicitudes (Grupo Fiancar)

Autor: Rodrigo Ilundain
Versión: V4 (completa)
Fecha de generación: 28/10/2025 20:57:12

Esta versión documenta el 100% de las funciones del backend (Code.gs) provisto, organizadas por módulos. Incluye firmas, código y notas de uso. El HTML del frontend no se incluye íntegro por extensión; se documentan sus interacciones.

1) Menú / WebApp

```
/* ====================== Menú / Web ====================== */
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('Solicitudes')
    .addItem('Iniciar solicitud', 'mostrarFormulario')
    .addToUi();
}

function doGet() {
  return HtmlService.createHtmlOutputFromFile('SolicitudAdelantoApp')
    .setTitle('Portal de Solicitudes')
    .setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL);
}

function mostrarFormulario() {
  const html = HtmlService.createHtmlOutputFromFile('SolicitudAdelantoApp')
    .setWidth(420)
    .setHeight(650);
  SpreadsheetApp.getUi().showModalDialog(html, 'Portal de Solicitudes');
}
```

2) Utilidades (fechas, propiedades, helpers)

```
/* ====================== Utilidades ====================== */
const SH_USUARIOS = 'Usuarios';
const PS = PropertiesService.getScriptProperties();
const cleanDigits_ = s => String(s||'').replace(/\D/g,'');
const getLibro = ()=> SpreadsheetApp.getActiveSpreadsheet();
const getHojaUsuarios = ()=> getLibro().getSheetByName(SH_USUARIOS);

// Fecha “ahora” sólida en la zona del script
function toUYDate_(d = new Date()){
  const tz = Session.getScriptTimeZone();
  const iso = Utilities.formatDate(d, tz, "yyyy-MM-dd'T'HH:mm:ss");
  return new Date(iso);
}
// STAMP dd/MM/yyyy HH:mm:ss
function stampNow_(){
  return Utilities.formatDate(toUYDate_(), Session.getScriptTimeZone(), "dd/MM/yyyy HH:mm:ss");
}

function monthSheetNameFrom_(date){
  const mes = date.toLocaleString('es-ES', { month: 'long' });
  const anio = date.getFullYear();
  return `${mes.charAt(0).toUpperCase()}${mes.slice(1)} ${anio}`;
}

const nowTs_ = ()=> Date.now();
const getJsonProp_ = (k,def=null)=>{ try{const v=PS.getProperty(k); return v?JSON.parse(v):def;}catch
const setJsonProp_ = (k,obj)=> PS.setProperty(k, JSON.stringify(obj));
```

```
/* Inserción protegida: nunca escribe en fila 1 */
function insertSafeRow_(hoja){
  const last = hoja.getLastRow();
  hoja.insertRowAfter(Math.max(1,last));
  return hoja.getLastRow(); // ≥ 2
}

/* === Parse seguro de fechas === */
function parseDateYMD_(s){
  if (s instanceof Date && !isNaN(s)) return new Date(s.getFullYear(), s.getMonth(), s.getDate());
  var str = String(s||'').trim();
  if (/^\d{4}-\d{2}-\d{2}$/.test(str)){ // YYYY-MM-DD
    var p = str.split('-'); return new Date(Number(p[0]), Number(p[1])-1, Number(p[2]));
  }
  if (/^\d{1,2}\/\d{1,2}\/\d{4}$/.test(str)){ // DD/MM/YYYY
    var q = str.split('/'); return new Date(Number(q[2]), Number(q[1])-1, Number(q[0]));
  }
  var d = new Date(str);
  return isNaN(d) ? new Date(NaN) : new Date(d.getFullYear(), d.getMonth(), d.getDate());
}
```

3) Conteo de días (A/B/C) y feriados

```
/* ====================== Conteo de días por categoría ====================== */
/*
   A: L-V (5 días, no cuenta sábados ni domingos)
   B: L-S (6 días). Si el rango incluye días L-V pero NO llega al sábado de esa semana,
      se agrega 1 sábado “fantasma” por cada semana parcial.
   C: L-V (igual que A)
   E: Reservada para Estudio (no se usa aquí).
*/
// Feriados fijos nacionales que NO cuentan como días de licencia
const FERIADOS_FIJOS_SET_ = new Set(["01-01","05-01","07-18","08-25","12-25"]); // MM-dd

function esFeriadoFijo_(d){
  return FERIADOS_FIJOS_SET_.has(Utilities.formatDate(d, Session.getScriptTimeZone(), "MM-dd"));
}

function daysCountByCategory_(f1, f2, categoria){
  const start = (f1 instanceof Date) ? new Date(f1.getFullYear(), f1.getMonth(), f1.getDate()) : par
  const end   = (f2 instanceof Date) ? new Date(f2.getFullYear(), f2.getMonth(), f2.getDate()) : par
  if (isNaN(start) || isNaN(end) || start > end) return 0;

  const cat = String(categoria||'B').trim().toUpperCase();

  if (cat === 'A' || cat === 'C'){
    let count = 0, d = new Date(start), endDay = new Date(end);
    while (d <= endDay){
      const dow = d.getDay();
      if (dow >= 1 && dow <= 5 && !esFeriadoFijo_(d)) count++;
      d.setDate(d.getDate()+1);
    }
    return count;
  }

  if (cat === 'B'){
    let count = 0, d = new Date(start), endDay = new Date(end);
    let sawWeekday = false, sawSaturday = false, curWeek = null;

    const weekKey = (dt)=>{
      const tmp = new Date(dt);
      const delta = 6 - tmp.getDay(); // hasta sábado
      tmp.setDate(tmp.getDate() + delta);
      return tmp.getFullYear() + '-' + (tmp.getMonth()+1) + '-' + tmp.getDate();
```

```
    };

    while (d <= endDay){
      const wk = weekKey(d);
      if (curWeek === null) curWeek = wk;
      if (wk !== curWeek){
        if (sawWeekday && !sawSaturday) count += 1; // sábado “fantasma”
        curWeek = wk; sawWeekday = false; sawSaturday = false;
      }

      const dow = d.getDay();
      if (dow !== 0 && !esFeriadoFijo_(d)){
        count++;
        if (dow >= 1 && dow <= 5) sawWeekday = true;
        if (dow === 6) sawSaturday = true;
      }
      d.setDate(d.getDate()+1);
    }
    if (sawWeekday && !sawSaturday) count += 1;
    return count;
  }
  return 0;
}
```

4) Encabezados, columnas y búsqueda por CI

```
/* ====================== Mapeo de encabezados ====================== */
function headerMap_(hoja){
  const lastCol = hoja.getLastColumn();
  const headers = hoja.getRange(1,1,1,lastCol).getDisplayValues()[0];
  const norm = s => String(s||'')
    .replace(/\u00A0/g,' ')
    .replace(/\s+/g,' ')
    .trim()
    .toUpperCase();
  const byName = {};
  headers.forEach((h,i)=> byName[norm(h)] = i+1);
  const findBy = (...labels) => {
    for (const lbl of labels){
      const c = byName[norm(lbl)];
      if (c) return c;
    }
    return null;
  };

  let col = {
    ACTIVIDAD: findBy('Actividad'),
    TOPE: findBy('Tope'),
    EMPRESA: findBy('Empresa'),
    FUNCIONARIO: findBy('Funcionario'),
    FECHA_PAGO: findBy('Fecha_pago','Fecha de pago','FECHA PAGO','Fecha Pago'),
    NOMBRE1: findBy('Nombre1','Nombre 1'),
    NOMBRE2: findBy('Nombre2','Nombre 2'),
    APELLIDO1: findBy('Apellido1','Apellido 1'),
    APELLIDO2: findBy('Apellido2','Apellido 2'),
    CI: findBy('CI','C.I.','Cedula','Cédula'),
    CEL: findBy('CEL','Celular','Telefono','Teléfono'),
    EMAIL: findBy('EMAIL','Mail','Correo'),
    CONCEPTO: findBy('CONCEPTO'),
    SECUENCIA: findBy('SECUENCIA','SEQUENCIA'),
    DEPTO: findBy('DEPTO','Departamento'),
    SUCURSAL: findBy('SUCURSAL','999'),
    SECCION: findBy('SECCION','Sección'),
    IMPORTE: findBy('IMPORTE'),
```

```
    SUCURSAL_NOMBRE: findBy('SUCURSAL_NOMBRE','Sucursal Nombre'),
    FECHA_REGISTRO: null,
    U_DIAS_FLEX: null,
    DIAS_LIC: null,
    EMAIL_TO: findBy('Emai gerente','EmaiL gerente','Email gerente'),
    EMAIL_CC: findBy('Email RRHH','Email RR.HH.','RRHH'),
    CATEGORIA_LIC: findBy('Categoria de licencia','Categoría de licencia','Calegoria de licencia','C
    SALARIO_TOPE: findBy('Tope salario vacacional','Tope vacacional','Tope salario','Topr vacacional
    DIAS_ESTUDIO: findBy('Dias por estudio','Días por estudio','DIAS POR ESTUDIO'),
    LIC_ESTUDIO_FLAG: findBy('Licencia por estudio act/des','Licencia por estudio ACT/DES','Licencia
  };

  // especiales (regex flexibles)
  for (const key in byName){
    if (/^REGISTRO\s*DE\s*FECHA$/i.test(key) || /^FECHA\s*DE\s*REGISTRO$/i.test(key) || /^REGISTRO\s
      col.FECHA_REGISTRO = byName[key];
    }
    if (/^LICENCIA\s+GENERADA/i.test(key)){
      col.U_DIAS_FLEX = byName[key];
    }
    if (/^DIAS?\s*DE\s*LICEN(C|S)IA$/i.test(key) || /^DIAS?_LIC$/i.test(key) || /^DIAS?\s*DISPONI\w+
      col.DIAS_LIC = byName[key];
    }
  }

  // Fallback por posición si faltan muchas
  const required = ['ACTIVIDAD','EMPRESA','FUNCIONARIO','NOMBRE1','APELLIDO1','CI','CEL','EMAIL','CO
  const missingCount = required.filter(k=>!col[k]).length;
  if (missingCount >= 5){
    const byPos = {
      ACTIVIDAD:1, TOPE:2, EMPRESA:3, FUNCIONARIO:4, FECHA_PAGO:5,
      NOMBRE1:6, NOMBRE2:7, APELLIDO1:8, APELLIDO2:9, CI:10, CEL:11, EMAIL:12,
      CONCEPTO:13, SECUENCIA:14, DEPTO:15, SUCURSAL:16, SECCION:17, IMPORTE:18,
      SUCURSAL_NOMBRE:19, FECHA_REGISTRO:20, U_DIAS_FLEX:21, DIAS_LIC:22,
      EMAIL_TO:23, EMAIL_CC:24, CATEGORIA_LIC:25, SALARIO_TOPE:27,
      DIAS_ESTUDIO:28, LIC_ESTUDIO_FLAG:29
    };
    Object.keys(byPos).forEach(k=>{
      if (!col[k] && lastCol >= byPos[k]) col[k] = byPos[k];
    });
  }

  const faltan = required.filter(k => !col[k]);
  if (faltan.length){
    throw new Error('Faltan columnas obligatorias en fila 1: ' + faltan.join(', '));
  }
  return col;
}

function findRowByCI_(hoja, ci){
  const cols = headerMap_(hoja);
  const cCI = cols.CI;
  const last = hoja.getLastRow();
  if (last < 2) return -1;
  const vals = hoja.getRange(2,cCI,last-1,1).getDisplayValues();
  const needle = cleanDigits_(ci);
  for (let i=0;i<vals.length;i++){
    if (cleanDigits_(vals[i][0]) === needle) return i+2;
  }
  return -1;
}
```

5) Saldos de licencia y estudio (leer/escribir)

```
/* Helpers DÍAS DISPONIBLES (licencia normal) */
function readDiasDisponibles_(hoja, fila, c){
  const colPref = c.DIAS_LIC || c.U_DIAS_FLEX;
  if (!colPref) return 0;
  return Number(hoja.getRange(fila, colPref).getValue()) || 0;
}
function writeDiasDisponibles_(hoja, fila, c, nuevoValor){
  const colPref = c.DIAS_LIC || c.U_DIAS_FLEX;
  if (!colPref) return;
  hoja.getRange(fila, colPref).setValue(nuevoValor);
}

/* Helpers DÍAS POR ESTUDIO */
function readDiasEstudio_(hoja, fila, c){
  if (!c.DIAS_ESTUDIO) return 0;
  return Number(hoja.getRange(fila, c.DIAS_ESTUDIO).getValue()) || 0;
}
function writeDiasEstudio_(hoja, fila, c, nuevoValor){
  if (!c.DIAS_ESTUDIO) return;
  hoja.getRange(fila, c.DIAS_ESTUDIO).setValue(nuevoValor);
}
```

6) Verificación de usuario y etiquetas U/V

```
/* ====================== Lógica de negocio ====================== */
function verificarUsuario(ciRaw){
  const hoja = getHojaUsuarios();
  if (!hoja) return { ok:false };
  const ci = cleanDigits_(ciRaw);
  if (!/^\d+$/.test(ci)) return { ok:false };
  const fila = findRowByCI_(hoja, ci);
  if (fila < 0) return { ok:false };

  const c = headerMap_(hoja);
  const diasLic = readDiasDisponibles_(hoja, fila, c);

  const datos = {
    habilitado: hoja.getRange(fila, c.ACTIVIDAD).getValue(),
    empresa: hoja.getRange(fila, c.EMPRESA).getDisplayValue(),
    funcionario: hoja.getRange(fila, c.FUNCIONARIO).getDisplayValue(),
    nombre1: hoja.getRange(fila, c.NOMBRE1).getDisplayValue(),
    nombre2: c.NOMBRE2 ? hoja.getRange(fila, c.NOMBRE2).getDisplayValue() : '',
    apellido1: hoja.getRange(fila, c.APELLIDO1).getDisplayValue(),
    apellido2: c.APELLIDO2 ? hoja.getRange(fila, c.APELLIDO2).getDisplayValue() : '',
    ci: cleanDigits_(hoja.getRange(fila, c.CI).getDisplayValue()),
    cel: hoja.getRange(fila, c.CEL).getDisplayValue(),
    email: hoja.getRange(fila, c.EMAIL).getDisplayValue(),
    secuencia: hoja.getRange(fila, (c.SEQUENCIA || c.SECUENCIA)).getDisplayValue(),
    depto: hoja.getRange(fila, c.DEPTO).getDisplayValue(),
    sucursal: hoja.getRange(fila, c.SUCURSAL).getDisplayValue(),
    seccion: hoja.getRange(fila, c.SECCION).getDisplayValue(),
    sucursal_nombre: hoja.getRange(fila, c.SUCURSAL_NOMBRE).getDisplayValue(),
    diasLicencia: diasLic,
    categoriaLic: c.CATEGORIA_LIC ? hoja.getRange(fila, c.CATEGORIA_LIC).getDisplayValue() : '',
    _fila: fila
  };
  return { ok:true, datos };
}

/* Etiquetas U/V */
function obtenerDatosLicencia(ciRaw){
  const hoja = getHojaUsuarios();
  if (!hoja) {
    return { diasDisponibles: 0, valorU: '', tituloU: 'Licencia Generada', tituloV: 'Días disponible
```

```
  }
  const fila = findRowByCI_(hoja, ciRaw);
  const c = headerMap_(hoja);
  const headerU = c.U_DIAS_FLEX ? (hoja.getRange(1, c.U_DIAS_FLEX).getDisplayValue() || 'Licencia Ge
  const headerV = c.DIAS_LIC ? (hoja.getRange(1, c.DIAS_LIC ).getDisplayValue() || 'Días disponibles
  if (fila < 0) {
    return { diasDisponibles: 0, valorU: '', tituloU: headerU, tituloV: headerV };
  }
  const dias = readDiasDisponibles_(hoja, fila, c);
  const valorU = c.U_DIAS_FLEX ? hoja.getRange(fila, c.U_DIAS_FLEX).getDisplayValue() : '';
  return { diasDisponibles: dias, valorU, tituloU: headerU, tituloV: headerV };
}

function obtenerDiasDisponibles(ciRaw){
  const hoja = getHojaUsuarios();
  if (!hoja) return 0;
  const fila = findRowByCI_(hoja, ciRaw);
  if (fila < 0) return 0;
  const c = headerMap_(hoja);
  return readDiasDisponibles_(hoja, fila, c);
}
```

7) Topes y Adelantos

```
/* ---------- Topes (B y AA) ---------- */
function obtenerTopes(ciRaw){
  const hoja = getHojaUsuarios();
  if (!hoja) return { ok:false, mensaje:"No existe la hoja 'Usuarios'." };
  const c = headerMap_(hoja);
  const ci = cleanDigits_(ciRaw);
  const fila = findRowByCI_(hoja, ci);
  if (fila < 0) return { ok:false, mensaje:"CI no encontrado." };

  let topeAd = c.TOPE ? Number(cleanDigits_(hoja.getRange(fila, c.TOPE).getDisplayValue())) : NaN;
  if (!Number.isFinite(topeAd) || topeAd <= 0) topeAd = 30000;

  let topeVac = c.SALARIO_TOPE ? Number(cleanDigits_(hoja.getRange(fila, c.SALARIO_TOPE).getDisplayV
  if (!Number.isFinite(topeVac)) topeVac = 0;

  return { ok:true, topeAdelanto: topeAd, topeVacacional: topeVac };
}

/* ---------- Adelantos ---------- */
function registrarSolicitud(usuario, celularIngresado, montoRaw){
  try{
    const libro = getLibro();
    const hojaUsuarios = getHojaUsuarios();
    if (!hojaUsuarios) return { ok:false, mensaje:"■ No existe la hoja 'Usuarios'." };
    const cU = headerMap_(hojaUsuarios);
    const ahora = toUYDate_();
    const dia = ahora.getDate();

    // Reglas de fecha para ADELANTO
    if (dia === 8 || dia === 9 || dia === 10 || dia >= 17 || dia === 1){
      return { ok:false, mensaje:"■ Fecha no válida. Contactar con RRHH." };
    }

    const montoStr = String(montoRaw||'').trim();
    if (!/^\d+$/.test(montoStr)) return { ok:false, mensaje:"■ Ingrese el monto solo con números." }
    const montoNum = parseInt(montoStr,10);
    if (montoNum<=0) return { ok:false, mensaje:"■ Importe inválido." };

    const fila = usuario._fila || findRowByCI_(hojaUsuarios, usuario.ci);
    if (fila < 0) return { ok:false, mensaje:"■ Usuario no encontrado." };
```

```
const hab = hojaUsuarios.getRange(fila, cU.ACTIVIDAD).getValue();
const habilitado = (hab===true) || ['true','1','si','sí','verdadero'].includes(String(hab).toLow
if (!habilitado) return { ok:false, mensaje:"■ Usted no está habilitado para adelantos." };

const celReal = cleanDigits_(hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue());
const celIng = cleanDigits_(celularIngresado);
const tailMatch = (a,b)=> !!a && !!b && (a.endsWith(b) || b.endsWith(a));
if (celIng.length<8 || !tailMatch(celReal, celIng)) return { ok:false, mensaje:"■ El celular no

const k = `last_adelanto_${usuario.ci}`;
const prev = getJsonProp_(k,{ts:0,monto:null});
if (prev && prev.monto==montoNum && (nowTs_()-Number(prev.ts))<30000){
  return { ok:false, mensaje:"■ Espere 30 segundos antes de reenviar." };
}

const nombreHoja = monthSheetNameFrom_(ahora);
let hoja = libro.getSheetByName(nombreHoja);
if (!hoja){
  hoja = libro.insertSheet(nombreHoja);
  hoja.appendRow(['Nombre1','Nombre2','Apellido1','Apellido2','CI','CEL','EMAIL','FECHA','Empres
}

const hmapMes = hoja.getRange(1,1,1,hoja.getLastColumn()).getDisplayValues()[0].reduce((acc,h,i)
const cCI_mes = hmapMes['CI'];
const datosMes = hoja.getDataRange().getValues();
let count=0;
for (let i=1;i<datosMes.length;i++){
  if (String(datosMes[i][cCI_mes-1])===String(usuario.ci)) count++;
}
if (count>=2) return { ok:false, mensaje:"■ Límite de 2 solicitudes por mes alcanzado." };

let topeRaw = cU.TOPE ? hojaUsuarios.getRange(fila, cU.TOPE).getDisplayValue() : '';
let topeNum = Number(cleanDigits_(topeRaw));
if (!Number.isFinite(topeNum) || topeNum <= 0) topeNum = 30000;
if (montoNum > topeNum) return { ok:false, mensaje:`■ El monto supera el tope por solicitud ($${

const diaNum = ahora.getDate();
const secuenciaValor = diaNum <= 9 ? '12' : (diaNum <= 19 ? '13' : '50');

hoja.appendRow([
  hojaUsuarios.getRange(fila, cU.NOMBRE1).getDisplayValue(),
  cU.NOMBRE2 ? hojaUsuarios.getRange(fila, cU.NOMBRE2).getDisplayValue() : '',
  hojaUsuarios.getRange(fila, cU.APELLIDO1).getDisplayValue(),
  cU.APELLIDO2 ? hojaUsuarios.getRange(fila, cU.APELLIDO2).getDisplayValue() : '',
  cleanDigits_(hojaUsuarios.getRange(fila, cU.CI).getDisplayValue()),
  hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.EMAIL).getDisplayValue(),
  Utilities.formatDate(ahora, Session.getScriptTimeZone(), "dd/MM/yyyy"),
  hojaUsuarios.getRange(fila, cU.EMPRESA).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.FUNCIONARIO).getDisplayValue(),
  '', '15', secuenciaValor,
  hojaUsuarios.getRange(fila, cU.DEPTO).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.SUCURSAL).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.SECCION).getDisplayValue(),
  montoNum,
  false,
  stampNow_()
]);

const colConf = hmapMes['CONFIRMACION'];
if (colConf) hoja.getRange(hoja.getLastRow(), colConf).insertCheckboxes().setValue(false);

setJsonProp_(k,{ts:nowTs_(),monto:montoNum});
return { ok:true, mensaje:"■ Adelanto registrado con éxito." };
```

```
  }catch(e){
    return { ok:false, mensaje:"■ Error registrando el adelanto: " + (e && e.message ? e.message : e
  }
}
```

8) Licencias normales (A/B/C)

```
/* ---------- Licencias (normal: A/B/C; E bloqueada aquí) ---------- */
function registrarLicencia(usuario, fechaInicio, fechaFin){
  const libro = getLibro();
  const hojaUsuarios = getHojaUsuarios();
  if (!hojaUsuarios) return { ok:false, mensaje:"■ No existe la hoja 'Usuarios'." };
  const cU = headerMap_(hojaUsuarios);
  const ahora = toUYDate_();
  const fila = usuario._fila || findRowByCI_(hojaUsuarios, usuario.ci);
  if (fila < 0) return { ok:false, mensaje:"■ Usuario no encontrado." };

  const k = `last_licencia_${usuario.ci}`;
  const prev = getJsonProp_(k,{ts:0});
  if (prev && (nowTs_()-Number(prev.ts))<30000){
    return { ok:false, mensaje:"■ Espere 30 segundos antes de enviar otra solicitud de licencia." };
  }

  const tz = Session.getScriptTimeZone();
  const fInicio = parseDateYMD_(fechaInicio);
  const fFin    = parseDateYMD_(fechaFin);
  if (isNaN(fInicio) || isNaN(fFin) || fInicio > fFin){
    return { ok:false, mensaje:"■ Rango de fechas inválido." };
  }

  const fHoy = parseDateYMD_(Utilities.formatDate(ahora, tz, "yyyy-MM-dd"));
  const ms = 24*60*60*1000;
  const diffDias = Math.round((fInicio - fHoy)/ms);
  if (!(diffDias >= 7)){
    return { ok:false, mensaje:"■ La solicitud debe ingresarse con al menos 7 días de anticipación."
  }

  const categoria = cU.CATEGORIA_LIC ? String(hojaUsuarios.getRange(fila, cU.CATEGORIA_LIC).getDispl
  if (categoria === 'E'){
    return { ok:false, mensaje:"■ La categoría E es exclusiva de Licencia por Estudio." };
  }

  let diasTomados = daysCountByCategory_(fInicio, fFin, categoria);
  if (!Number.isFinite(diasTomados) || diasTomados<=0){
    return { ok:false, mensaje:"■ Rango sin días contables." };
  }

  const saldo = readDiasDisponibles_(hojaUsuarios, fila, cU);
  if (diasTomados > saldo) return { ok:false, mensaje:"■ No tiene suficientes días disponibles." };
  writeDiasDisponibles_(hojaUsuarios, fila, cU, saldo - diasTomados);

  registrarFechaTimestamp_(hojaUsuarios, fila, cU);

  const nombreHoja = `Licencia ${monthSheetNameFrom_(fInicio)}`;
  let hojaMes = libro.getSheetByName(nombreHoja);
  if (!hojaMes){
    hojaMes = libro.insertSheet(nombreHoja);
    hojaMes.appendRow(['Actividad','Nombre1','Nombre2','Apellido1','Apellido2','CI','CEL','EMAIL','E
  }

  hojaMes.appendRow([
    hojaUsuarios.getRange(fila, cU.ACTIVIDAD).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.NOMBRE1).getDisplayValue(),
    cU.NOMBRE2 ? hojaUsuarios.getRange(fila, cU.NOMBRE2).getDisplayValue() : '',
```

```
    hojaUsuarios.getRange(fila, cU.APELLIDO1).getDisplayValue(),
    cU.APELLIDO2 ? hojaUsuarios.getRange(fila, cU.APELLIDO2).getDisplayValue() : '',
    cleanDigits_(hojaUsuarios.getRange(fila, cU.CI).getDisplayValue()),
    hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.EMAIL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.EMPRESA).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.FUNCIONARIO).getDisplayValue(),
    Utilities.formatDate(ahora, tz, "dd/MM/yyyy"),
    '263',
    hojaUsuarios.getRange(fila, (cU.SEQUENCIA || cU.SECUENCIA)).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.DEPTO).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SUCURSAL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SECCION).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.IMPORTE).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SUCURSAL_NOMBRE).getDisplayValue(),
    Utilities.formatDate(fInicio, tz, "dd/MM/yyyy"),
    Utilities.formatDate(fFin, tz, "dd/MM/yyyy"),
    diasTomados,
    '',
    false,
    categoria,
    stampNow_()
  ]);

  const hmapMes = hojaMes.getRange(1,1,1,hojaMes.getLastColumn()).getDisplayValues()[0].reduce((acc,
  const colConf = hmapMes['CONFIRMACION']; if (colConf) hojaMes.getRange(hojaMes.getLastRow(), colCo
  let colStamp = hmapMes['STAMP'];
  if (!colStamp) {
    colStamp = hojaMes.getLastColumn() + 1;
    hojaMes.insertColumnAfter(hojaMes.getLastColumn());
    hojaMes.getRange(1, colStamp).setValue('STAMP');
  }
  hojaMes.getRange(hojaMes.getLastRow(), colStamp).setValue(stampNow_());

  // Emails (TO/CC) omitidos aquí por brevedad del bloque (idénticos al original)
  setJsonProp_(k,{ts:nowTs_()});
  return { ok:true, mensaje:`■ Licencia registrada. Categoría ${categoria} aplicada.` };
}
```

9) Salario vacacional

```
/* ---------- Salario Vacacional ---------- */
function isVacacionalHabilitado_() {
  const hoja = getHojaUsuarios();
  if (!hoja) return false;
  try {
    const v = hoja.getRange(2, 26).getValue(); // Z2 por diseño previo
    return v === true || String(v).toLowerCase() === 'true';
  } catch (e) { return false; }
}
function obtenerFlagVacacional() { return isVacacionalHabilitado_(); }

function registrarSalarioVacacional(usuario, celularIngresado, montoRaw){
  try{
    if (!isVacacionalHabilitado_()) {
      return { ok:false, mensaje:"■ El adelanto por salario vacacional no está habilitado." };
    }
    const libro = getLibro();
    const hojaUsuarios = getHojaUsuarios();
    if (!hojaUsuarios) return { ok:false, mensaje:"■ No existe la hoja 'Usuarios'." };
    const cU = headerMap_(hojaUsuarios);
    const ahora = toUYDate_();

    const montoStr = String(montoRaw||'').trim();
```

```
if (!/^\d+$/.test(montoStr)) return { ok:false, mensaje:"■ Ingrese el monto solo con números." }
const montoNum = parseInt(montoStr,10);
if (montoNum<=0) return { ok:false, mensaje:"■ Importe inválido." };

const fila = usuario._fila || findRowByCI_(hojaUsuarios, usuario.ci);
if (fila < 0) return { ok:false, mensaje:"■ Usuario no encontrado." };

const hab = hojaUsuarios.getRange(fila, cU.ACTIVIDAD).getValue();
const habilitado = (hab===true) || ['true','1','si','sí','verdadero'].includes(String(hab).toLow
if (!habilitado) return { ok:false, mensaje:"■ Usted no está habilitado para adelantos." };

const celReal = cleanDigits_(hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue());
const celIng = cleanDigits_(celularIngresado);
const tailMatch = (a,b)=> !!a && !!b && (a.endsWith(b) || b.endsWith(a));
if (celIng.length<8 || !tailMatch(celReal, celIng)) return { ok:false, mensaje:"■ El celular no

const k = `last_vacacional_${usuario.ci}`;
const prev = getJsonProp_(k,{ts:0,monto:null});
if (prev && prev.monto==montoNum && (nowTs_()-Number(prev.ts))<30000){
  return { ok:false, mensaje:"■ Espere 30 segundos antes de reenviar." };
}

let topeVacRaw = cU.SALARIO_TOPE ? hojaUsuarios.getRange(fila, cU.SALARIO_TOPE).getDisplayValue(
let topeVacNum = Number(cleanDigits_(topeVacRaw));
if (!Number.isFinite(topeVacNum)) topeVacNum = 0;
if (topeVacNum <= 0) return { ok:false, mensaje:"■ Sin tope habilitado para salario vacacional."
if (montoNum > topeVacNum) return { ok:false, mensaje:`■ El monto supera el tope ($${topeVacNum}

const diaNum = ahora.getDate();
const secuenciaValor = diaNum <= 9 ? '12' : (diaNum <= 19 ? '13' : '50');

const nombreHoja = `Salario vacacional ${monthSheetNameFrom_(ahora)}`;
let hoja = libro.getSheetByName(nombreHoja);
if (!hoja){
  hoja = libro.insertSheet(nombreHoja);
  hoja.appendRow(['Nombre1','Nombre2','Apellido1','Apellido2','CI','CEL','EMAIL','FECHA','Empres
}

hoja.appendRow([
  hojaUsuarios.getRange(fila, cU.NOMBRE1).getDisplayValue(),
  cU.NOMBRE2 ? hojaUsuarios.getRange(fila, cU.NOMBRE2).getDisplayValue() : '',
  hojaUsuarios.getRange(fila, cU.APELLIDO1).getDisplayValue(),
  cU.APELLIDO2 ? hojaUsuarios.getRange(fila, cU.APELLIDO2).getDisplayValue() : '',
  cleanDigits_(hojaUsuarios.getRange(fila, cU.CI).getDisplayValue()),
  hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.EMAIL).getDisplayValue(),
  Utilities.formatDate(ahora, Session.getScriptTimeZone(), "dd/MM/yyyy"),
  hojaUsuarios.getRange(fila, cU.EMPRESA).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.FUNCIONARIO).getDisplayValue(),
  '', '256', secuenciaValor,
  hojaUsuarios.getRange(fila, cU.DEPTO).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.SUCURSAL).getDisplayValue(),
  hojaUsuarios.getRange(fila, cU.SECCION).getDisplayValue(),
  montoNum,
  false,
  stampNow_()
]);

const hmap = hoja.getRange(1,1,1,hoja.getLastColumn()).getDisplayValues()[0].reduce((acc,h,i)=>{
const colConf = hmap['CONFIRMACION'];
if (colConf) hoja.getRange(hoja.getLastRow(), colConf).insertCheckboxes().setValue(false);

setJsonProp_(k,{ts:nowTs_(),monto:montoNum});
return { ok:true, mensaje:"■ Adelanto de salario vacacional registrado con éxito." };
```

```
  }catch(e){
    return { ok:false, mensaje:"■ Error registrando el adelanto (vacacional): " + (e && e.message ?
  }
}
```

10) Wrappers públicos para el frontend

```
/* ---------- WRAPPERS front (adelanto / licencia / vacacional) ---------- */
function enviarAdelanto(payload){
  try{
    const ci = cleanDigits_(payload && payload.ci);
    const monto = payload && payload.monto;
    const celular = payload && payload.celular;
    if (!/^\d+$/.test(String(ci))) return { ok:false, mensaje:"■ Ingrese CI sin puntos ni guiones."
    const ver = verificarUsuario(ci);
    if (!ver.ok) return { ok:false, mensaje:"■ CI no encontrado." };
    return registrarSolicitud(ver.datos, celular, monto);
  }catch(e){
    return { ok:false, mensaje:"■ Error: " + (e && e.message ? e.message : e) };
  }
}
function enviarLicencia(payload){
  try{
    const ci = cleanDigits_(payload && payload.ci);
    const f1 = payload && payload.fechaDesde;
    const f2 = payload && payload.fechaHasta;
    if (!/^\d+$/.test(String(ci))) return { ok:false, mensaje:"■ Ingrese CI sin puntos ni guiones."
    const ver = verificarUsuario(ci);
    if (!ver.ok) return { ok:false, mensaje:"■ CI no encontrado." };
    return registrarLicencia(ver.datos, f1, f2);
  }catch(e){
    return { ok:false, mensaje:"■ Error: " + (e && e.message ? e.message : e) };
  }
}
function enviarSalarioVacacional(payload){
  try{
    const ci = cleanDigits_(payload && payload.ci);
    const monto = payload && payload.monto;
    const celular = payload && payload.celular;
    if (!/^\d+$/.test(String(ci))) return { ok:false, mensaje:"■ Ingrese CI sin puntos ni guiones."
    const ver = verificarUsuario(ci);
    if (!ver.ok) return { ok:false, mensaje:"■ CI no encontrado." };
    return registrarSalarioVacacional(ver.datos, celular, monto);
  }catch(e){
    return { ok:false, mensaje:"■ Error: " + (e && e.message ? e.message : e) };
  }
}
```

11) Historiales (adelantos y licencias)

```
/* ---------- Historiales ---------- */
function obtenerHistorial(ciRaw){
  const libro = getLibro();
  const hojas = libro.getSheets();
  const historial = [];
  const tz = Session.getScriptTimeZone();
  const ci = cleanDigits_(ciRaw);
  hojas.forEach(hoja=>{
    const nombre = hoja.getName();
    if (/^[A-ZÁÉÍÓÚÑ][a-záéíóúñ]+ \d{4}$/.test(nombre)){
      const datos = hoja.getDataRange().getValues();
      if (datos.length < 2) return;
      const hmap = hoja.getRange(1,1,1,hoja.getLastColumn()).getDisplayValues()[0].reduce((acc,h,i)=
      const cCI = hmap['CI'], cFECHA = hmap['FECHA'], cIMP = hmap['IMPORTE'];
```

```
      if (!cCI || !cFECHA || !cIMP) return;
      for (let i=1;i<datos.length;i++){
        if (String(datos[i][cCI-1]) == String(ci)){
          const fr = datos[i][cFECHA-1];
          const fecha = (fr instanceof Date && !isNaN(fr)) ? fr : new Date(String(fr));
          const monto = datos[i][cIMP-1];
          if (!isNaN(fecha) && monto !== '' && monto != null){
            historial.push({ fecha: Utilities.formatDate(fecha,tz,"dd/MM/yyyy"), tipo:'Adelanto', va
          }
        }
      }
    }
  });
  historial.sort((a,b)=>{
    const da = new Date(a.fecha.split('/').join('-'));
    const db = new Date(b.fecha.split('/').join('-'));
    return db - da;
  });
  return historial.slice(0,8);
}

function obtenerHistorialLicencias(ciRaw){
  const libro = getLibro();
  const hojas = libro.getSheets();
  const tz = Session.getScriptTimeZone();
  const ci = cleanDigits_(ciRaw);
  const res = [];
  hojas.forEach(hoja=>{
    const nombre = hoja.getName();
    if (!/^Licencia\s+/i.test(nombre)) return;
    const lastRow = hoja.getLastRow();
    if (lastRow < 2) return;
    const headers = hoja.getRange(1,1,1,hoja.getLastColumn()).getDisplayValues()[0];
    const hmap = headers.reduce((acc,h,i)=>{ acc[String(h).toUpperCase()] = i+1; return acc; },{});
    const cCI = hmap['CI'];
    const cF1 = hmap['FECHA INICIO'];
    const cF2 = hmap['FECHA FIN'];
    const cDIAS = hmap['DIAS TOMADOS'];
    const cCAT = hmap['CATEGORIA'] || hmap['CATEGORÍA'];
    const cCONF = hmap['CONFIRMACION'] || hmap['CONFIRMACIÓN'];
    if (!cCI || !cF1 || !cF2 || !cDIAS) return;
    const values = hoja.getRange(2,1,lastRow-1,hoja.getLastColumn()).getValues();
    values.forEach(row=>{
      if (String(row[cCI-1]) == String(ci)){
        const f1 = row[cF1-1], f2 = row[cF2-1];
        const dias = row[cDIAS-1];
        const cat = cCAT ? row[cCAT-1] : '';
        const conf = cCONF ? row[cCONF-1] : '';
        const fmt = v=>{
          if (v instanceof Date && !isNaN(v)) return Utilities.formatDate(v,tz,"dd/MM/yyyy");
          const d = new Date(v);
          return isNaN(d) ? String(v||'') : Utilities.formatDate(d,tz,"dd/MM/yyyy");
        };
        res.push({
          fechaInicio: fmt(f1), fechaFin: fmt(f2),
          dias: Number(dias)||0, categoria: String(cat||''),
          confirmacion: (conf===true || String(conf).toLowerCase()==='true') ? '✔' : ''
        });
      }
    });
  });
  res.sort((a,b)=>{
    const da = new Date(a.fechaFin.split('/').join('-'));
    const db = new Date(b.fechaFin.split('/').join('-'));
```

```
    return db - da;
  });
  return res.slice(0,20);
}
```

12) Registro de usuario (validaciones y timestamp)

```
/* ---------- Registro de nuevo usuario ---------- */
function registrarNuevoUsuario(datos){
  const hoja = getHojaUsuarios();
  if (!hoja) return { ok:false, mensaje:"■ No existe la hoja 'Usuarios'." };
  const c = headerMap_(hoja);

  // unicidad
  const data = hoja.getDataRange().getValues();
  const ciNew = cleanDigits_(datos.ci);
  const celNew = cleanDigits_(datos.cel);
  for (let i=1;i<data.length;i++){
    if (c.CI && cleanDigits_(data[i][c.CI-1]) == ciNew) return { ok:false, mensaje:"■ CI ya existe."
    if (c.CEL && cleanDigits_(data[i][c.CEL-1]) == celNew) return { ok:false, mensaje:"■ Celular ya
    if (c.EMAIL && String(data[i][c.EMAIL-1]).toLowerCase() == String(datos.email).toLowerCase()){
      return { ok:false, mensaje:"■ Email ya registrado." };
    }
  }

  const row = insertSafeRow_(hoja);
  const setIf = (col,val)=>{ if (col) hoja.getRange(row,col).setValue(val); };

  setIf(c.EMPRESA, datos.empresa || '');
  setIf(c.FUNCIONARIO, datos.funcionario || '');
  setIf(c.FECHA_PAGO, '');
  setIf(c.NOMBRE1, datos.nombre1 || '');
  setIf(c.NOMBRE2, datos.nombre2 || '');
  setIf(c.APELLIDO1, datos.apellido1 || '');
  setIf(c.APELLIDO2, datos.apellido2 || '');
  setIf(c.CI, ciNew || '');
  setIf(c.CEL, "'" + (celNew || ''));
  setIf(c.EMAIL, datos.email || '');
  setIf(c.CONCEPTO, '15');
  setIf(c.SECUENCIA, '');
  setIf(c.DEPTO, '999');
  setIf(c.SUCURSAL, '999');
  setIf(c.SECCION, '99999');
  setIf(c.IMPORTE, '');
  setIf(c.SUCURSAL_NOMBRE, datos.sucursal || '');
  if (c.FECHA_REGISTRO) setIf(c.FECHA_REGISTRO, Utilities.formatDate(new Date(), Session.getScriptTi

  if (c.ACTIVIDAD){
    hoja.getRange(row, c.ACTIVIDAD).insertCheckboxes();
    hoja.getRange(row, c.ACTIVIDAD).setValue(true);
  }
  if (c.CATEGORIA_LIC && !String(hoja.getRange(row,c.CATEGORIA_LIC).getDisplayValue()).trim()){
    hoja.getRange(row,c.CATEGORIA_LIC).setValue('B');
  }

  registrarFechaTimestamp_(hoja, row, c);
  return { ok:true, mensaje:"■ Usuario registrado con éxito (habilitado para adelantos)." };
}
```

13) Licencia por Estudio (flag, saldo, registro y wrapper)

```
/* ====================== LICENCIA POR ESTUDIO ====================== */
function obtenerFlagLicEstudio(ciRaw){
  const hoja = getHojaUsuarios();
```

```
  if (!hoja) return false;
  const c = headerMap_(hoja);
  const fila = findRowByCI_(hoja, cleanDigits_(ciRaw));
  if (fila < 0) return false;
  if (!c.LIC_ESTUDIO_FLAG) return false;
  const v = hoja.getRange(fila, c.LIC_ESTUDIO_FLAG).getValue();
  return v === true || String(v).toLowerCase() === 'true';
}

function obtenerDatosLicenciaEstudio(ciRaw){
  const hoja = getHojaUsuarios();
  if (!hoja) return { diasDisponibles: 0 };
  const c = headerMap_(hoja);
  const fila = findRowByCI_(hoja, cleanDigits_(ciRaw));
  if (fila < 0) return { diasDisponibles: 0 };
  const dias = readDiasEstudio_(hoja, fila, c);
  return { diasDisponibles: dias, titulo: 'Días por estudio' };
}

function contarFeriadosFijosEnRango_(ini, fin){
  let d = new Date(ini.getFullYear(), ini.getMonth(), ini.getDate());
  const end = new Date(fin.getFullYear(), fin.getMonth(), fin.getDate());
  let c = 0;
  while (d <= end){
    if (esFeriadoFijo_(d)) c++;
    d.setDate(d.getDate()+1);
  }
  return c;
}

function registrarLicenciaEstudio(usuario, fechaInicio, fechaFin){
  const libro = getLibro();
  const hojaUsuarios = getHojaUsuarios();
  if (!hojaUsuarios) return { ok:false, mensaje:"■ No existe la hoja 'Usuarios'." };
  const cU = headerMap_(hojaUsuarios);
  const fila = usuario._fila || findRowByCI_(hojaUsuarios, usuario.ci);
  if (fila < 0) return { ok:false, mensaje:"■ Usuario no encontrado." };

  if (!cU.LIC_ESTUDIO_FLAG || !(hojaUsuarios.getRange(fila, cU.LIC_ESTUDIO_FLAG).getValue()===true |
      String(hojaUsuarios.getRange(fila, cU.LIC_ESTUDIO_FLAG).getValue()).toLowerCase()==='true')){
    return { ok:false, mensaje:"■ La licencia por estudio no está habilitada para su usuario." };
  }

  if (!fechaFin) fechaFin = fechaInicio;

  const tz = Session.getScriptTimeZone();
  const ahora = toUYDate_();
  const fInicio = parseDateYMD_(fechaInicio);
  const fFin = parseDateYMD_(fechaFin);
  const fHoy = parseDateYMD_(Utilities.formatDate(ahora, tz, "yyyy-MM-dd"));
  const ms = 24*60*60*1000;
  if (isNaN(fInicio) || isNaN(fFin) || fInicio > fFin){
    return { ok:false, mensaje:"■ Rango de fechas inválido." };
  }
  const diffDias = Math.round((fInicio - fHoy)/ms);
  if (!(diffDias >= 7)){
    return { ok:false, mensaje:"■ Debe solicitarse con al menos 7 días de anticipación." };
  }

  const di = new Date(fInicio.getFullYear(), fInicio.getMonth(), fInicio.getDate());
  const df = new Date(fFin.getFullYear(), fFin.getMonth(), fFin.getDate());
  let diasTomados = Math.round((df - di)/ms) + 1;
  const feriadosEnRango = contarFeriadosFijosEnRango_(di, df);
  diasTomados = Math.max(0, diasTomados - feriadosEnRango);
```

```
  if (diasTomados <= 0) return { ok:false, mensaje:"■ El rango sólo contiene feriados/no contables."
  if (diasTomados > 9) return { ok:false, mensaje:"■ Máximo 9 días por solicitud de estudio." };

  const saldoEst = readDiasEstudio_(hojaUsuarios, fila, cU);
  if (diasTomados > saldoEst) return { ok:false, mensaje:"■ No tiene suficientes días por estudio di

  writeDiasEstudio_(hojaUsuarios, fila, cU, saldoEst - diasTomados);
  registrarFechaTimestamp_(hojaUsuarios, fila, cU);

  const nombreHoja = `Licencia ${monthSheetNameFrom_(fInicio)}`;
  let hojaMes = libro.getSheetByName(nombreHoja);
  if (!hojaMes){
    hojaMes = libro.insertSheet(nombreHoja);
    hojaMes.appendRow(['Actividad','Nombre1','Nombre2','Apellido1','Apellido2','CI','CEL','EMAIL','E
  }

  hojaMes.appendRow([
    hojaUsuarios.getRange(fila, cU.ACTIVIDAD).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.NOMBRE1).getDisplayValue(),
    cU.NOMBRE2 ? hojaUsuarios.getRange(fila, cU.NOMBRE2).getDisplayValue() : '',
    hojaUsuarios.getRange(fila, cU.APELLIDO1).getDisplayValue(),
    cU.APELLIDO2 ? hojaUsuarios.getRange(fila, cU.APELLIDO2).getDisplayValue() : '',
    cleanDigits_(hojaUsuarios.getRange(fila, cU.CI).getDisplayValue()),
    hojaUsuarios.getRange(fila, cU.CEL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.EMAIL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.EMPRESA).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.FUNCIONARIO).getDisplayValue(),
    Utilities.formatDate(toUYDate_(), Session.getScriptTimeZone(), "dd/MM/yyyy"),
    '100E',
    hojaUsuarios.getRange(fila, (cU.SEQUENCIA || cU.SECUENCIA)).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.DEPTO).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SUCURSAL).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SECCION).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.IMPORTE).getDisplayValue(),
    hojaUsuarios.getRange(fila, cU.SUCURSAL_NOMBRE).getDisplayValue(),
    Utilities.formatDate(fInicio, Session.getScriptTimeZone(), "dd/MM/yyyy"),
    Utilities.formatDate(fFin, Session.getScriptTimeZone(), "dd/MM/yyyy"),
    diasTomados,
    '',
    false,
    'E',
    stampNow_()
  ]);

  const hmapMes = hojaMes.getRange(1,1,1,hojaMes.getLastColumn()).getDisplayValues()[0].reduce((acc,
  const colConf = hmapMes['CONFIRMACION']; if (colConf) hojaMes.getRange(hojaMes.getLastRow(), colCo
  let colStamp = hmapMes['STAMP'];
  if (!colStamp) {
    colStamp = hojaMes.getLastColumn() + 1;
    hojaMes.insertColumnAfter(hojaMes.getLastColumn());
    hojaMes.getRange(1, colStamp).setValue('STAMP');
  }
  hojaMes.getRange(hojaMes.getLastRow(), colStamp).setValue(stampNow_());
  return { ok:true, mensaje:`■ Licencia por estudio registrada (${diasTomados} día/s netos).` };
}

/* Wrapper expuesto al front */
function enviarLicenciaEstudio(payload){
  try{
    const ci = cleanDigits_(payload && payload.ci);
    const f1 = payload && payload.fechaDesde;
    let f2 = payload && payload.fechaHasta;
    if (!/^\d+$/.test(String(ci))) return { ok:false, mensaje:"■ Ingrese CI sin puntos ni guiones."
    if (!f2) f2 = f1;
```

```
    const ver = verificarUsuario(ci);
    if (!ver.ok) return { ok:false, mensaje:"■ CI no encontrado." };
    return registrarLicenciaEstudio(ver.datos, f1, f2);
  }catch(e){
    return { ok:false, mensaje:"■ Error: " + (e && e.message ? e.message : e) };
  }
}
```

14) Helper de sello de fecha/hora

```
/* ====== Helper de timestamp reutilizable ====== */
function registrarFechaTimestamp_(hoja, fila, c) {
  try {
    if (c && c.FECHA_REGISTRO) {
      hoja.getRange(fila, c.FECHA_REGISTRO).setValue(stampNow_());
    }
  } catch(e) { Logger.log('Error en registrarFechaTimestamp_: ' + e); }
}
```

Notas finales

• Todos los bloques de código aquí incluidos corresponden al Code.gs provisto y han sido organizados por módulos para facilitar su lectura y mantenimiento. • Las plantillas de email (HTML) y el frontend (HTML/CSS/JS) interactúan mediante google.script.run con los wrappers expuestos.