

FELIZ NO SIMPLES aka AGORA VAI

1) [config.gs](#)

```
/**  
  
 * config.gs  
  
 * Variáveis e inicialização  
  
 */  
  
const SPREADSHEET_GDO_SIGILAO =  
  '1-WHrgPTL7vD21a8PY5Nhea7Jm6JbPLYJ-IJaX2XwNng'; // ⚠ CONFIRME se  
  está correto  
  
const ABA_CATALOGO = 'BOTA_TUDO';  
  
const ABA_RIFAS = 'RIFA_SIGILAO';  
  
const ABA_CHECKOUT = 'CHECKOUT';  
  
const ABA_FAVORITAS = 'FAVORITAS';  
  
  
// GitHub  
  
const GITHUB_REPO =  
  PropertiesService.getScriptProperties().getProperty('GITHUB_REPO') ||  
  'BetShopTV/betshoptv.github.io';  
  
const GITHUB_BRANCH =  
  PropertiesService.getScriptProperties().getProperty('GITHUB_BRANCH')  
  || 'main';  
  
const GITHUB_FILE_PATH =  
  PropertiesService.getScriptProperties().getProperty('GITHUB_FILE_PATH'  
  ') || 'dados/gdo-bstv.json';
```

```
const GITHUB_FILE_PATH_RIFAS = 'dados/rifa-bstv.json'; //  NOVO
arquivo pro JSON de rifas

const GITHUB_TOKEN =
PropertiesService.getScriptProperties().getProperty('GITHUB_TOKEN')
|| 'NAO PODE MANDAREEEEE MAS EU SEI QUALÉ E TÁ CERTO JURO';

// Base do site / imagens

const SITE_BASE_URL =
PropertiesService.getScriptProperties().getProperty('SITE_BASE_URL')
|| 'https://betshoptv.com';

const IMAGE_BASE_URL =
PropertiesService.getScriptProperties().getProperty('IMAGE_BASE_URL')
|| SITE_BASE_URL + '/img/';

// Código inicial para codigo_login (três dígitos, sem "CR", começa
em 030)

const START_CODIGO_NUM =
Number(PropertiesService.getScriptProperties().getProperty('START_COD
IGO_NUM')) || 30;
```

2) [utils.gs](#)

```
/** endpoint.gs - robusto e compatível
```

```

    * Usa as constantes de config.gs (SPREADSHEET_GDO_SIGILAO,
    ABA_CHECKOUT, ABA_FAVORITAS, etc.)

    */

/* doGet: aceita fallback via Image() (GET) para quando o frontend
usa a "imagem ping" */

function doGet(e) {

    try {

        Logger.log("doGet received: " + JSON.stringify(e.parameter ||
{}));

        // transforma query params em payload e encaminha

        const payload = Object.assign({}, e.parameter || {});

        // se items vier como string JSON, tenta parsear

        if (payload.items && typeof payload.items === 'string' &&
payload.items.trim().startsWith('[')) {

            try { payload.items = JSON.parse(payload.items); } catch(err){}

        }

        // decide e encaminha

        if (isFavoritePayload(payload)) {

            return gravarFavorito(payload);

        } else {

            return gravarCheckout(payload);

        }

    } catch (err) {

        Logger.log("doGet error: " + err);
    }
}

```

```

        return
    ContentService.createTextOutput(JSON.stringify({status:'error',
message: String(err)}))

        .setMimeType(ContentService.MimeType.JSON);

    }

}

/* doPost: aceita JSON, form-data e urlencoded */

function doPost(e) {

    try {

        Logger.log("=== doPost START ===");

        // Para debug: log raw postData se existir

        if (e.postData) {

            Logger.log("postData.type: " + (e.postData.type || ''));

            Logger.log("postData.contents: " + (e.postData.contents ||
''));

        }

        // Monta payload de forma resiliente

        let payload = {};

        if (e.postData && e.postData.contents) {

            const t = (e.postData.type || '').toLowerCase();

            if (t.indexOf('json') > -1) {

                try { payload = JSON.parse(e.postData.contents); }

                catch(err){ payload = e.parameter || {}; }

            } else {

```

```
        payload = e.parameter || {};\n\n    }\n\n    } else {\n\n        payload = e.parameter || {};\n\n    }\n\n\n    // se veio embrulhado em { payload: {...} }\n\n    if (payload && payload.payload) payload = payload.payload;\n\n\n\n    // se items chegou como string JSON (via FormData), converte\n\n    if (payload.items && typeof payload.items === 'string' &&\n        payload.items.trim().startsWith('[')) {\n\n        try { payload.items = JSON.parse(payload.items); } catch(err){\n            Logger.log('items parse err: '+err); }\n\n    }\n\n\n\n    Logger.log("Parsed payload: " + JSON.stringify(payload));\n\n\n\n    // Decide FAVORITAS x CHECKOUT\n\n    if (isFavoritePayload(payload)) {\n\n        return gravarFavorito(payload);\n\n    } else {\n\n        return gravarCheckout(payload);\n\n    }\n\n}
```

```

    } catch (err) {

        Logger.log("doPost error: " + err);

        return
        ContentService.createTextOutput(JSON.stringify({status:'error',
message: String(err)}))

        .setMimeType(ContentService.MimeType.JSON);

    }
}

/* Helper: detectar favorite */

function isFavoritePayload(payload) {

    if (!payload) return false;

    const a = (payload.action || payload._action ||
    '').toString().toLowerCase();

    if (a === 'favorite' || a === 'fav' || a === 'favorite_add') return
true;

    if (payload.fav_foi || payload.fav || payload.favorite === '1' ||
payload.favorite === true) return true;

    return false;
}

/* GRAVAR FAVORITO (FAVORITAS) */

function gravarFavorito(payload) {

    const ss = SpreadsheetApp.openById(SPREADSHEET_GDO_SIGILAO);

```

```

const sh = ss.getSheetByName(ABA_FAVORITAS);

if (!sh) throw new Error('Aba FAVORITAS não encontrada');

const headers = readHeaders(sh);

const row = headers.map(h => {

    const low = (h || '').toLowerCase();

    if (low === 'timestamp') return new Date();

    if (low === 'id_pecas' || low === 'pecas_foi' || low === 'fav_foi')
return payload.pecas_foi || payload.id_pecas || payload.id || '';

    if (low === 'codigo_login') return payload.codigo_login ||
payload.codigologin || payload.codigo || '';

    if (low === 'dispositivo') return payload.dispositivo ||
payload.device || '';

    return payload[h] || payload[low] || '';

});

sh.appendRow(row);

Logger.log("Favorito gravado: " + JSON.stringify(row));

return ContentService.createTextOutput(JSON.stringify({status:'ok',
destino:'FAVORITAS'}))

    .setMimeType(ContentService.MimeType.JSON);
}

/* GRAVAR CHECKOUT (CHECKOUT) - grava uma linha consolidada por
compra */

```

```
function gravarCheckout(payload) {

    const ss = SpreadsheetApp.openById(SPREADSHEET_GDO_SIGILAO);

    const sh = ss.getSheetByName(ABA_CHECKOUT);

    if (!sh) throw new Error('Aba CHECKOUT não encontrada');

    const headers = readHeaders(sh);

    // NORMALIZAÇÃO: items (array) / id_peca (string) / id_peca
    separado por ; , |

    let items = [];

    if (Array.isArray(payload.items)) {

        items = payload.items;

    } else if (payload.items && typeof payload.items === 'string') {

        // tenta parse JSON

        if (payload.items.trim().startsWith('[')) {

            try { items = JSON.parse(payload.items); } catch(err){ items =
String(payload.items).split(/[,;|]+/).map(x=>x.trim()).filter(Boolean
); }

        } else {

            items =
String(payload.items).split(/[,;|]+/).map(x=>x.trim()).filter(Boolean
);

        }

    } else if (payload.id_peca) {

        // caso id_peca seja um único id
```



```

    items =
String(payload.id_peca).split(/[;,|]+/).map(x=>x.trim()).filter(Boole
an);

    } else if (payload.items_str) {

        items =
String(payload.items_str).split(/[;,|]+/).map(x=>x.trim()).filter(Boo
lean);

    }

    // codigo_login: reaproveita se enviado, senão pede
nextCodigoLogin() do utils.gs (se existir)

    let codigo = payload.codigo_login || payload.codigologin ||
payload.codigo || '';

    if (!codigo) {

        try {

            // nextCodigoLogin() no seu utils devolve algo tipo '030' -
acrescentamos 'CR'

            codigo = 'CR' + nextCodigoLogin();

        } catch(err) {

            // fallback timestamp-based

            codigo = 'CR' + Utilities.formatDate(new Date(), "GMT-3",
"yyMMddHHmmssSSS");

        }

    }

    // preco_total: prioriza preco_total, depois preco_pago /
preco_peca, ou soma se payload.precos[] existir

```

```
    let preco_total = payload.preco_total || payload.preco_pago ||
payload.total || '';

    if (!preco_total && Array.isArray(payload.precos)) {

        preco_total =
payload.precos.reduce((s,v)=>s+Number(v||0),0).toFixed(2);

    }

    // agora montamos o objeto normalizado para mapear nas colunas

    const norm = {

        codigo_login: codigo,

        nome: payload.nome || payload.name || '',

        email: payload.email || payload.mail || '',

        telefone: payload.telefone || payload.phone || payload.whats ||
'',

        peca_foi: items.join(','),

        rifa_foi: payload.rifa_foi || payload.id_rifa || '',

        fav_peca: payload.fav_peca || '',

        preco_total: (preco_total !== undefined ? preco_total : ''),

        cidade: payload.cidade || payload.city || '',

        pais: payload.pais || payload.country || 'BR',

        fonte_utm: payload.fonte_utm || payload.utm_source || '',

        dispositivo: payload.dispositivo || payload.device || (typeof
navigator !== 'undefined'?navigator.userAgent:''),

        formapagamento: payload.formapagamento || payload.FormaPagamento
|| payload.forma || payload.payment || 'PIX'
```

```
});

// monta a linha conforme cabeçalhos

const row = headers.map(h => {

  const low = (h || '').toLowerCase();

  if (low === 'timestamp') return new Date();

  if (low === 'codigo_login') return norm.codigo_login;

  if (low === 'nome') return norm.nome;

  if (low === 'email') return norm.email;

  if (low === 'telefone') return norm.telefone;

  if (low === 'peca_foi') return norm.peca_foi;

  if (low === 'rifa_foi') return norm.rifa_foi;

  if (low === 'fav_peca') return norm.fav_peca;

  if (low === 'preco_total') return norm.preco_total;

  if (low === 'cidade') return norm.cidade;

  if (low === 'pais') return norm.pais;

  if (low === 'fonte_utm') return norm.fonte_utm;

  if (low === 'dispositivo') return norm.dispositivo;

  if (low === 'formapagamento') return norm.formapagamento;

  // fallback general: tenta pegar payload por header exato

  return payload[h] || payload[low] || '';

});
```

```

        sh.appendRow(row);

        Logger.log("Checkout gravado: codigo=" + codigo + " items=" +
JSON.stringify(items) + " row=" + JSON.stringify(row));

        return ContentService.createTextOutput(JSON.stringify({status:'ok',
destino:'CHECKOUT', codigo_login: codigo}))

        .setMimeType(ContentService.MimeType.JSON);
    }
}

```

3) [catalogo.gs](#)

```

/**

 * utils.gs

 * Helpers

 */

function nowSP() {

    const d = new Date();

    return Utilities.formatDate(d, "GMT-3", "dd-MM-yyyy HH:mm:ss"); //
    ✓ alterado

}

```

```

function nextCodigoLogin() {

    const ss = SpreadsheetApp.openById(SPREADSHEET_GDO_SIGILAO);

    const sh = ss.getSheetByName(ABA_CHECKOUT);

    if (!sh) return String(START_CODIGO_NUM).padStart(3, '0');

    const data = sh.getRange(2, 2, Math.max(0, sh.getLastRow() - 1),
1).getValues()

        .filter(r => r && r[0])

        .map(r => String(r[0]).replace(/\D/g, ''))

        .filter(Boolean)

        .map(Number)

        .sort((a, b) => a - b);

    let last = START_CODIGO_NUM;

    if (data.length) last = Math.max(last, data[data.length - 1]);

    return String(last + 1).padStart(3, '0');

}

```

```

function readHeaders(sheet) {

    const h = sheet.getRange(1, 1, 1,
sheet.getLastColumn()).getValues()[0] || [];

    return h.map(x => (x == null ? '' : String(x).trim()));

}

```

```

function montarUrlsImagens(arquivosImg) {

```

```

const base = IMAGE_BASE_URL.replace(/\/+$/, '') + '/';

let arr = [];

if (!arquivosImg) return arr;

if (Array.isArray(arquivosImg)) arr = arquivosImg;

else arr = String(arquivosImg).split(/[,;|\s]+/).map(x =>
x.trim()).filter(Boolean);

const unique = Array.from(new Set(arr));

return unique.map(code => ({

  tmb: base + 'tmb/' + code + '.avif',

  otm: base + 'otm/' + code + '.avif',

  fallback: base + 'fallback/' + code + '.jpg',

  code: code

})));
}

```

4) [github.gs](#)

```

/**
 * catalogo.gs

```

```

* Gera JSON público de produtos e rifas

*/

/** =====

* 🖼️ JSON principal de produtos (só com arquivos_img preenchido)

* ===== */

function gerarJSON() {

    const ss = SpreadsheetApp.openById(SPREADSHEET_GDO_SIGILAO);

    const sh = ss.getSheetByName(ABA_CATALOGO);

    if (!sh) throw new Error("Aba catálogo não encontrada");

    const data = sh.getDataRange().getValues();

    const headers = data.shift(); // tira a primeira linha (cabeçalho)

    // Índice da coluna img_peca (coluna D = índice 3)

    const idxImg = 3;

    // filtra: só linhas com img_peca preenchido

    const produtos = data

        .filter(r => r[idxImg] && String(r[idxImg]).trim() !== "")

        .map(r => {

            const obj = {};

            headers.forEach((h, i) => obj[h] = r[i]);

```

```

const arquivos = obj["img_peca"] || ""; // nome da coluna D

const imagens = montarUrlsImagens(arquivos);


return {

    id: obj["id_peca"] || "",

    grupo: obj["grupo"] || "",

    peca: obj["peca"] || "",

    titulo_completo: obj["titulo_completo"] || "",

    titulo: obj["titulo_completo"] || "", //  compatibilidade

    tecnica: obj["tecnica"] || "",

    dimensao: obj["dimensao"] || "",

    preco: Number(obj["preco_peca"]) || 0,

    status: obj["status_peca"] || "",

    arquivos_img: arquivos,

    imagens

};

});

return ContentService.createTextOutput(

    JSON.stringify({produtos}, null, 2)

).setMimeType(ContentService.MimeType.JSON);

}

```



```

/** =====

* 📄 JSON separado para RIFAS

* ===== */

function gerarJSONRifas() {

  const ss = SpreadsheetApp.openById(SPREADSHEET_GDO_SIGILAO);

  const sh = ss.getSheetByName(ABA_RIFAS);

  if (!sh) throw new Error('Aba RIFA_SIGILAO não encontrada');

  const data = sh.getDataRange().getValues();

  if (!data || data.length < 2) return { atualizado: nowSP(),
totalRifas: 0, rifas: [] };

  const headers = data[0].map(h => String(h || '').trim());

  const rows = data.slice(1);

  const rifas = rows

    .map(r => {

      const obj = {};

      headers.forEach((col, i) => obj[col] = r[i]);

      return obj;

    })

    .filter(obj => obj['id_rifa'] && String(obj['id_rifa']).trim()
!= '')

    .map(obj => ({

```

```

        id_rifa: obj['id_rifa'] || '',

        hex_rifa: obj['hex_rifa'] || '',

        botao_rifa: obj['botao_rifa'] || '',

        status_rifa: obj['status_rifa'] || ''

    }));

const jsonFinal = {

    atualizado: nowSP(),

    totalRifas: rifas.length,

    rifas

};

const jsonStr = JSON.stringify(jsonFinal, null, 2);

publicarNoGitHub(jsonStr, GITHUB_FILE_PATH_RIFAS);

return jsonFinal;
}

/** =====

 * 🔍 doGet - debug manual

 * ===== */

function doGet(e) {

    try {

        const tipo = (e && e.parameter && e.parameter.tipo) || 'pecas';

```

```

    const data = tipo === 'rifas' ? gerarJSONRifas() : gerarJSON();

    return
    ContentService.createTextOutput(JSON.stringify(data)).setMimeType(ContentService.MimeType.JSON);

  } catch (err) {

    return ContentService.createTextOutput(JSON.stringify({ erro:
err.message })).setMimeType(ContentService.MimeType.JSON);

  }
}

```

5) [trigger.gs](#)

```

/**
 * github.gs
 * Publica JSONs no GitHub (UTF-8 fix definitivo)
 */

function publicarNoGitHub(jsonStr, filePath) {

  const token = GITHUB_TOKEN;

  const path = filePath || GITHUB_FILE_PATH;

```

```
const url = 'https://api.github.com/repos/' + GITHUB_REPO +
'/contents/' + path;

const sha = obterShaAtual(url, token);

// ⚙️ conversão explícita UTF-8 → Base64

const blob = Utilities.newBlob('', 'application/json',
'data.json');

blob.setDataFromString(jsonStr, 'UTF-8');

const utf8Base64 = Utilities.base64Encode(blob.getBytes());

const payload = {

  message: 'Atualização automática ' + nowSP(),

  content: utf8Base64,

  branch: GITHUB_BRANCH

};

if (sha) payload.sha = sha;

const options = {

  method: 'put',

  contentType: 'application/json',

  payload: JSON.stringify(payload),

  headers: {

    Authorization: 'Bearer ' + token,

    'User-Agent': 'AppsScript',
```

```

        'Accept': 'application/vnd.github+json'

    },

    muteHttpExceptions: true

};

const res = UrlFetchApp.fetch(url, options);

Logger.log('GitHub update code: ' + res.getResponseCode() + ' → ' +
path);

return res.getContentText();
}

function obterShaAtual(url, token) {

    try {

        const res = UrlFetchApp.fetch(url, {

            method: 'get',

            headers: { Authorization: 'Bearer ' + token, 'User-Agent':
'AppsScript' }

        });

        const data = JSON.parse(res.getContentText());

        return data.sha;

    } catch {

        return null;

    }

}

```

6) endpoint.gs

```

/*****

🕒 Atualizações automáticas

*****/

// ✅ Executa a cada 15 minutos o ciclo completo (catálogo + rifas)

function criarTriggerCatalogo() {

    ScriptApp.newTrigger('atualizaPeriodicamente')

        .timeBased()

        .everyMinutes(15)

        .create();

}

/**

 * Atualiza JSONs quando houver edição manual na aba BOTA_TUDO

 */

function onEdit(e) {
```

```
try {

    const sh = e.range.getSheet();

    if (sh.getName() === ABA_CATALOGO) gerarJSON();

} catch (err) {

    Logger.log('onEdit erro: ' + err.message);

}

}

/**

 * Atualiza os dois JSONs no GitHub (catálogo e rifas)

 */

function atualizaPeriodicamente() {

    gerarJSON();          // catálogo

    gerarJSONRifas();     // rifas

}
```