



DOSSIÊ BSTV – ALINHAMENTO

GERAL (v10.2025)



0] VERDADES ATUAIS DO PROJETO

0a] O núcleo de dados do BetShopTV.com vive em GDO_SIGILAO.

0b] As colunas e nomes das abas devem ser sempre seguidas à risca.

0c] JSONs publicados pelo Apps Script vão para o GitHub repo
→ [BetShopTV/betshoptv.github.io/dados/](https://betshopTV/betshoptv.github.io/dados/)

0d] Política de imagens (imutável)

- .avif: /img/tmb/ e /img/otm/

- .jpg: /img/fallback/ (último caso)

0e] O sistema PIX e rifas do site é a referência. Nada precisa ser refeito – só replicado o mesmo esquema para o novo fluxo de compras da Central de Apostas.



1] ESTRUTURA DA PLANILHA “GDO_SIGILAO”

1a] Abas ativas e função

ABA	FUNÇÃO
OTA_TUDO	atálogo de peças completo (dados visuais e técnicos).
IFA_SIGILAO	dados internos das rifas (hex, botões, status, login).
HECKOUT	registros de compras (sessões finalizadas, PIX, contatos).
ESSAO	sessões ativas ou encerradas (peças, rifas, likes, tempo).

ECAS_JSON criação do `pecas.json` pro GitHub.

IFAS_JSON criação do `rifas.json` pro GitHub.

ENDAS_OFERTAS / BOTA_FORA logs administrativos e backups de venc

⚠️ A aba *FAVORITAS* foi oficialmente desativada.
Agora tudo que era “curtidas” e “favoritos” está
centralizado em **SESSAO**.

2] COLUNAS-CHAVE (respeitar nomes e grafias)

2a] CHECKOUT

```
timestamp | id_login | nome | email | telefone | pecas | rifas |  
favoritas | total
```

👉 usado pra registrar o “carrinho finalizado” de cada sessão.

2b] SESSAO

```
timestamp | session_id | pecas | rifas | favoritas | duracao |  
finalizou
```

👉 representa o comportamento e métrica da sessão.
likes, cliques, tempo e encerramento são salvos aqui.

2c] RIFA_SIGILAO

```
id_rifa | hex_rifa | botao_rifa | status_rifa | codigo_login
```

👉 base interna para montar o front de rifas (grid colorida de 170 botões).

2d] PECAS_JSON

```
id_peca | titulo | preco | status | tecnica | dimensao | imagens
```

👉 usado diretamente pelo script que gera `/dados/pecas.json`.

2e] RIFAS_JSON

`id_rifa | hex | numero | status_rifa`

👉 gera `/dados/rifas.json`.

⚙️ 3] ARQUITETURA DE FUNCIONAMENTO

3a] Pipeline geral

SHEETS (SIGILÃO)

↓

GAS (gera e publica JSONs)

↓

GitHub `/dados/*.json`

↓

SITE (HTML/JS ou Next.js)

3b] JSONs ativos

Arquivo	Origem	Atualiza por	Função
<code>ecas.json</code>	ECAS_JSON	AS (trigger)	atualiza galeria de peças
<code>ifas.json</code>	IFAS_JSON	AS (trigger)	atualiza lista de rifas
<code>estatisticas.json</code>	futuro	anual/cron	estatísticas gerais

📄 4] FLUXOS DO SITE → PLANILHA

🛒 4a] A. Central de Apostas (Checkout PIX)

O que acontece:

quando o usuário encerra o “carrinho”, o front soma todos os

valores (peças + rifas) e envia um **POST** pro Apps Script com o mesmo formato da página de rifas.

Campos obrigatórios (respeitar nomes da planilha)

Campo	Tipo	Origem	Exemplo
<code>id_login</code>	string	front	<code>R042</code>
<code>peças</code>	string	front	<code>AC01,AC02"</code>
<code>rifas</code>	string	front	<code>R001,R004"</code>
<code>favoritas</code>	number	front	(likes na sessão)
<code>total</code>	number	oma da sessão	<code>85.50</code>
<code>nome, email, telefone</code>	string	ormulário (opcional)	
<code>timestamp</code>	gerado pelo GAS	<code>2025-10-12 23:58</code>	

Comportamento do GAS

- Gera o `id_login` se não houver.
 - Garante que o **total do PIX** = valor do carrinho.
 - Gera payload BRCODE e exibe **QR + Pix Copia e Cola** (como nas rifas).
 - Registra a compra em **CHECKOUT**.
- **Front:** só precisa repetir a lógica da página de rifas.
- **GAS:** já possui função `gravarCheckout(payload)` que mapeia esses campos.
- **Importante:** não gerar QR no backend – o site faz isso.

💖 4b] B. Likes por Sessão

Função: registrar **quantos likes totais** o visitante deu naquela sessão.

Como funciona:

1. O site mantém contador local (`sessionStorage`).

Ao finalizar o carrinho **ou ao sair**, envia ao GAS:

```
{
  "session_id": "sess_20251012_041",
  "favoritas": 7,
  "pecas": "AC01,AC02",
  "rifas": "R003",
  "duracao": "00:04:37",
  "finalizou": true
}
```

2.

3. O GAS escreve linha em **SESSAO** com esses dados.

4. Assim é possível rastrear **atividade geral**, **engajamento** e **tempo médio** sem precisar de tabela de favoritos separada.

● *benefício:* simples, leve e compatível com o modelo de contagem “fidedigno o suficiente”.



5] JSONs PUBLICADOS NO GITHUB

✓ Todos devem ir pra pasta:

betshoptv.github.io/dados/

JSON	Função	Origem	Atualização
dados/pecas.json	atálogo de peça	PECAS_JS	origiger GAS

`dados/rifas.json` rid de rifas ba `RIFAS_JS` trigger GAS
`dados/sessao.json` estatísticas ba `SESSAO` anual/cron
`futuro)`






● Todos são publicados via **GitHub API (contents:write)** usando token armazenado nas Script Properties.

6] RESUMO DE TAREFAS PRO OUTRO GPT

✓ 0 QUE JÁ EXISTE

- Sistema PIX dinâmico (funcional na página de rifas).
- Scripts GAS que geram e publicam JSONs.
- Estrutura de planilha consolidada (campos e nomes).

⚙️ 0 QUE PRECISA FAZER

1.  Garantir que `rifas.json` e `pecas.json` são atualizados juntos pelo mesmo gatilho.
2.  Reutilizar **lógica PIX das rifas** para a **Central de Apostas** (checkout total).
3.  Adicionar **POST de sessão** (likes, duração, finalizou) → aba `SESSAO`.
4.  Manter toda a nomenclatura idêntica à planilha (sem variações de nome).
5.  Publicar todos os JSONs sempre em `/dados/`.

7] PRÓS & CONSIDERAÇÕES

Prós

estrutura modular (GAS + Sheetstensão total à grafia dos nomes de
itHub)

colunas

SONs simples e fáceis de consuuidado com cache dos fetch no fro


tualizações automáticas por vitar colunas extras não nomeadas
rigger “Unnamed”)


ontagem de likes simplificada vitar duplicar registros de sessã

8] TOM E CONTEXTO ARTÍSTICO

BSTV é um sistema performático de arte e transação, que se
equilibra entre loja, instalação e circo de espelhos.

O objetivo não é só vender, mas tensionar os sistemas de
valor e troca – transformar o ato de clicar, comprar e curtir
em gestos performativos dentro do universo BSTV.

 a estética da gambiarra não é um tema central – é método.

 o projeto vive no improviso, mas busca rigor na estrutura.