Stack 100% gratis (sin tarjeta, sin Blaze)

Hosting (frontend)

- **Cloudflare Pages (Free)
    - Deploy automático desde GitHub (CI/CD).
    - Límite Free: hasta 20,000 archivos por sitio.

Backend (datos + tiempo real + auth)

- **Firebase Authentication (Spark)
- **Cloud Firestore (Spark)
    - Realtime listeners para standings y bracket (público).
    - En Spark, si te pasas de cuota, simplemente deja de servir y devuelve "resource exhausted" (no te cobra).

Lo que NO usaremos (porque rompe el "100% gratis")

- **Cloud Functions**: la doc oficial indica upgrade a Blaze para desplegar funciones.
- **Cloud Storage for Firebase**: cambios anunciados (sept 2024) y requerimientos posteriores; para no arriesgarte a bloqueos/billing, evitamos Storage y PDFs alojados allí.

Arquitectura técnica (end-to-end)

Frontend (SPA)

Opciones

- JS puro (rápido y suficiente) o React/Vite (más ordenado).
- Recomendación: **Vite + React** por mantenibilidad, pero si ya traes Vanilla, se puede.

Módulos UI

1. **Landing pública del evento**
    - /e/:eventId
    - Tabs: Standings (realtime), Bracket (realtime), Equipos, Reglas/Horarios

2. **Auth**
    - Registro/login email+password (gratis en Auth).
3. **Dashboard usuario**
    - “Mi equipo”, invitaciones, estado de participación.
    - “Mis certificados” (participación y winners si aplica).
4. **Dashboard admin (solo admins)**
    - Cerrar registro
    - Generar bracket (dinámico)
    - Confirmar winners (1°, 2°, 3°)
    - Emitir certificados (participación y winners)

Realtime

- onSnapshot() a:
    - events/{eventId}
    - matches filtrados por eventId
    - standings/{eventId} o standings_items por eventId

Backend (Firestore: colecciones y documentos)

events/{eventId}

- name, date
- status: registro_abierto | cerrado | en_curso | finalizado
- format: single_elim
- seeding_mode: manual | random | ranking
- winners (manual por admin):
    - winners_confirmed: boolean
    - first_team_id, second_team_id, third_team_id
    - winners_confirmed_by, winners_confirmed_at

teams/{teamId}

- event_id
- name
- leader_user_id
- seed (opcional)

team_members/{docId}

- event_id
- team_id
- user_id
- role: leader | member
- invite_status: invited | accepted | rejected

brackets/{eventId}

- event_id
- size (8/16/32…)
- rounds
- locked: boolean

matches/{matchId}

- event_id
- round (1..R)
- index (posición en ronda)
- teamA_id, teamB_id
- winner_id
- status: draft | confirmed | auto_advanced
- next_match_id
- next_slot: A|B

Standings (dos enfoques)

A) Documento cacheado

- standings/{eventId}: lista ordenada y compacta (para lectura rápida pública)

B) Colección de items

- standings_items/{docId} con event_id, team_id, points, time_ms, etc.

Lógica crítica (sin Cloud Functions)

Como no usamos Functions (para seguir 100% gratis), toda la lógica "de servidor" se ejecuta en el **cliente admin** (con reglas fuertes y transacciones).

1) Generación de bracket dinámico (admin)

- Admin presiona "Generar bracket".
- El frontend:
    1. Lee teams del evento.
    2. Calcula P = nextPowerOfTwo(N) y BYEs.
    3. Crea brackets/{eventId}.
    4. Escribe todos los matches con:
        - round, index
        - next_match_id, next_slot
        - asignación inicial de equipos en R1
    5. Ejecuta auto-advance de BYEs (set winner_id + propagar al siguiente match).

Notas técnicas

- Firestore batch: máximo 500 writes por batch (fragmentar si hace falta).
- Para N≤100, bracket size típico 128 (127 matches) => cabe bien.

2) Confirmación de resultados (staff/admin)

- En cada match, admin selecciona ganador y confirma:
    - set winner_id, status=confirmed
    - escribe el ganador en next_match_id en el slot correspondiente (A/B)

Esto debe ser:

- Transacción (para evitar estados inválidos).
- Validaciones UI (no permitir confirmar si faltan equipos).

3) Winners manuales (admin)

- Admin elige team_id para 1°, 2°, 3°.
- Guarda en events/{eventId} winners_* y winners_confirmed=true.

Certificados PDF 100% gratis (sin Storage)

Tu idea de "plantillas HTML" es correcta, pero para mantenerlo 100% gratis:

✅ **Estrategia**

- **No guardas el PDF** en la nube.
- Guardas solo el **registro del certificado** (metadatos) en Firestore.
- El PDF se **genera on-demand en el navegador** y se descarga al momento.

Esto cumple:

- "Se asocia a sus cuentas" → porque el usuario ve sus certificados en su panel (desde Firestore).
- "Exporta automáticamente a PDF" → usando librería client-side.

Plantillas

- participacion.html
- winner_1.html
- winner_2.html
- winner_3.html

Generación automática en frontend

- Librería recomendada: **html2pdf.js** (html2canvas + jsPDF) (todo local, gratis).
- Alternativa: window.print() (menos "automático", depende del usuario).

Datos (todos desde Firestore)

- nombre_usuario
- team_name
- event_name, event_date
- type (participación o winner_1/2/3)
- folio (certificate_id)
- QR opcional con URL de verificación (también generado client-side)

Modelo Firestore para certificados

certificates/{certificateId}

- event_id
- user_id
- type: participation | winner_1 | winner_2 | winner_3
- place: null|1|2|3
- issued_at
- issued_by (admin uid)
- status: issued | revoked

Regla anti-duplicados

- Único por (event_id, user_id, type):
    - Antes de crear, consultas si ya existe.
    - Si admin corrige winners: revocas y re-emites.

Emisión

- **Participación**: admin pulsa “Emitir participación”
    - Crea certificados para todos los team_members accepted del evento.
- **Winners**: admin confirma winners y pulsa “Emitir winners”
    - Crea certificados winner_* para miembros de esos equipos.

Seguridad (sin custom claims, 100% Spark)

Como no tienes Functions para asignar roles con custom claims, la solución gratis y práctica es:

Admin por UID "whitelist"

- En un doc config/admins guardas lista de UIDs (o lo pones en rules).
- Firestore Rules: solo esos UIDs pueden:
    - cerrar registro
    - generar bracket
    - confirmar resultados
    - set winners
    - emitir certificados

Esto es suficiente para un evento pequeño (<100).

Despliegue (gratis)

1. Repo en GitHub
2. Cloudflare Pages conecta el repo y despliega
3. Firebase:
    - Auth habilitado (Email/Password)
    - Firestore (Spark)
    - Reglas y índices listos

Checklist técnico por áreas

Frontend

- SPA + rutas públicas/privadas
- Auth + estado de sesión
- Realtime bracket + standings

- Admin panel (acciones write)
- Motor PDF:
    - render plantilla HTML
    - inyectar datos
    - export pdf con html2pdf
    - botón “Descargar”

Firestore

- Estructura de colecciones
- Índices para queries por event_id
- Reglas:
    - público: read de eventos/matches/standings
    - usuarios: read de sus certificados + su membership
    - admin: write de matches/winners/certificates/brackets

Recomendación final para mantener “100% gratis”

- Firestore = fuente de verdad (todo).
- PDFs = generación local (no Storage).
- Admin UID whitelist para permisos.
- Cloudflare Pages para hosting.

Si quieres, en el siguiente paso te dejo:

1. las **reglas de Firestore** (listas para pegar), y
2. el **plan de carpetas** del frontend + los “services” de Firestore (queries y writes) para bracket, winners y certificados.