SpiritPlanner

Herramienta para gestionar campañas en solitario de **Spirit Island** usando **Firestore** como fuente de verdad.

Permite generar campañas completas, jugar incursiones con control de tiempo por sesiones, calcular puntuaciones y exportar resultados a **TSV** para análisis externo (Google Sheets / BGG).

Objetivo

- Gestionar **Eras**, **Periodos** e **Incursiones**
- Jugar Incursiones en orden libre dentro de cada Periodo
- Registrar tiempo real mediante sesiones (pausa/reanudación)
- Calcular puntuaciones de forma consistente
- Exportar resultados a TSV

Fuera de alcance cualquier funcionalidad no relacionada con lo anterior.

Stack tecnológico

- **Firestore (Spark)** → fuente única de verdad
- **Android (Flet)** → UI principal (lectura/escritura)
- **PC (scripts / CLI Python)** → generación de Era y exportación TSV
- **Google Sheets** → diario y preparación manual para BGG

Regla global de idioma (CRÍTICA)

- **Código, modelos, campos, colecciones, scripts → INGLÉS**
- **Interfaz de usuario → CASTELLANO**

Nunca mezclar idiomas en nombres técnicos.

Modelo de datos (Firestore)

Estructura general

```
eras/{era_id}
  periods/{period_id}
    incursions/{incursion_id}
      sessions/{session_id}
```

Era (`eras/{era_id}`)

- `is_active` (bool)
- `created_at` (timestamp UTC)

Estado derivado

- `active_incursion` (object | null)
  - `period_id` (string)
  - `incursion_id` (string)

Reglas:

- `active_incursion` identifica la Incursión activa actual
- Se escribe al iniciar una Incursión
- Se elimina al finalizar una Incursión
- Es un estado derivado para optimización; Firestore sigue siendo la única fuente de verdad

---

Period (`periods/{period_id}`)

- `index` (int)
- `created_at` (timestamp UTC)
- `revealed_at` (timestamp | null)
- `adversaries_assigned_at` (timestamp | null)
- `ended_at` (timestamp | null)

Significado de estados:

- `revealed_at`:
  - marca el **inicio real del Periodo**
- `adversaries_assigned_at`:

    - confirma que los adversarios han sido asignados correctamente
- `ended_at`:
    - el Periodo está completamente finalizado

Reglas:

- Los Periodos se revelan **secuencialmente**
- Solo se puede revelar un Periodo si el anterior está finalizado
- La asignación de adversarios solo es posible tras revelar el Periodo
- Una vez `adversaries_assigned_at != null`, la asignación queda bloqueada

Incursion (`incursions/{incursion_id}`)

Identidad

- `index` (int)

Setup base (solo PC, inmutable)

- `spirit_1_id` (string)
- `spirit_2_id` (string)
- `board_1` (string)
- `board_2` (string)
- `board_layout` (string)

Asignación estratégica

- `adversary_id` (string | null)

Reglas de asignación (a nivel de Periodo):

- Cada Periodo tiene **exactamente 4 Incursiones**
- Las 4 deben tener `adversary_id` no nulo
- Los 4 `adversary_id` deben ser **distintos**
- `"scenario"` cuenta como adversario válido
- La asignación queda bloqueada tras `adversaries_assigned_at`

Al iniciar la Incursión

- `adversary_level` (string | null)
- `difficulty` (int)

Notas:

- Los escenarios se tratan como adversarios
- El nivel es texto libre

Estado temporal

- `started_at` (timestamp | null)
- `ended_at` (timestamp | null)

Reglas:

- Solo puede existir **una Incursión activa** en toda la Era
- Las Incursiones de un Periodo pueden jugarse en cualquier orden

Score

- `result` (`"win"` / `"loss"`)
- `player_count` (int)
- `invader_cards_remaining` (int)
- `invader_cards_out_of_deck` (int)
- `dahan_alive` (int)
- `blight_on_island` (int)
- `score` (int)

Reglas:

- El score se calcula **al finalizar**
- El score es **inmutable**

Exportación

- `exported` (bool)

Sessions (`sessions/{session_id}`)

- `started_at` (timestamp UTC)
- `ended_at` (timestamp | null)

Reglas:

- Múltiples sesiones por Incursión
- Solo una sesión abierta
- No se borran sesiones desde Android

- Duración total = suma de sesiones

Reglas de escritura

Generar Era (PC)

- Crea Era, Periodos e Incursions
- Solo setup base
- No escribe campos de estado

Revelar Periodo (Android)

- Fija `revealed_at`

Asignar adversarios (Android)

- Escribe `adversary_id` en las 4 Incursiones
- Valida reglas
- Si es correcto:
  - fija `adversaries_assigned_at`

Iniciar Incursión (Android)

- Valida:
  - Periodo revelado
  - adversarios asignados
  - no hay Incursión activa
- Fija `started_at`
- Asigna `adversary_level`
- Calcula `difficulty`
- Registra Incursión activa en la Era

Finalizar Incursión (Android)

- Fija `ended_at`
- Calcula `score`
- Elimina Incursión activa
- Finaliza el Periodo si procede
- El formulario de resultado y puntuación se muestra al iniciar la acción de finalizar, no como estado previo.

PC puede modificar cualquier campo sin restricciones.

Manejo de tiempo y UI (ACLARACIÓN IMPORTANTE)

- Firestore **almacena siempre timestamps absolutos (UTC)**
- **Nunca** se ajusta manualmente DST en persistencia
- La **conversión a hora local se hace SOLO en la UI**
- La UI debe:
  - usar la zona horaria real del dispositivo
  - aplicar correctamente reglas históricas de DST
- El formato de presentación es:
  - `dd/mm/yy HH:MM`

Esta regla es de **presentación**, no de modelo de datos.

Navegación y vistas

1. Eras
2. Periodos
3. Incursiones
4. Incursión (sesiones y resultado)

Reglas de UI:

- Flujo guiado por **un único botón por Periodo**
- El botón representa el estado
- No hay textos explícitos de estado
- Restricción dura: una Incursión activa

Puntuación

Victoria:

- `5 × difficulty`
- `+10`
- `+2 × invader_cards_remaining`

Derrota:

- `2 × difficulty`
- `+1 × invader_cards_out_of_deck`

Siempre:

- `+ player_count × dahan_alive`
- `− player_count × blight_on_island`

Exportación TSV

- Una fila = Incursión finalizada
- Solo `ended_at != null`
- Archivo sobrescrito
- Orden: Era → Periodo → Incursión
- Timestamps ISO 8601 UTC

Estado del diseño

- Modelo: cerrado
- Reglas: cerradas
- Flujo: coherente
- Listo para implementación