

Análisis y Diseño del Sistema: Gestor de Torneos

Documentación de Proyecto

26 de septiembre de 2025

Índice

1. Introducción	2
2. Objetivos del Sistema	2
3. Arquitectura del Sistema	2
4. Diseño de la Interfaz de Usuario (UI)	3
4.1. Etapa 1: Equipos	3
4.2. Etapa 2: Grupos	3
4.3. Etapa 3: Eliminatorias	5
5. Diseño de la Base de Datos	5
6. Consideraciones Técnicas y Decisiones de Diseño	6

1. Introducción

Este documento explica de manera sencilla cómo está pensado y diseñado el **Gestor de Torneos**, una aplicación de escritorio para administrar torneos de fútbol de principio a fin. El programa guía al usuario por tres etapas principales:

- Etapa 1: Equipos
- Etapa 2: Grupos
- Etapa 3: Eliminatorias

La idea es que cualquiera pueda entender rápidamente cómo funciona y qué hace cada parte del sistema.

2. Objetivos del Sistema

Los principales objetivos que la aplicación busca cumplir son:

- **Gestión de Equipos:** Permitir al usuario crear, modificar y eliminar equipos, asignándolos a zonas o grupos predefinidos.
- **Automatización del Fixture:** Generar automáticamente los enfrentamientos para la fase de grupos (“todos contra todos”) y las fases eliminatorias.
- **Registro de Resultados:** Ofrecer una interfaz clara para que el usuario ingrese los resultados de los partidos.
- **Cálculo de Posiciones:** Calcular y mostrar en tiempo real la tabla de posiciones de la fase de grupos, ordenada por puntos y criterios de desempate.
- **Flujo de Torneo Guiado y Flexible:** Asegurar que el usuario no pueda avanzar a una etapa sin haber completado los requisitos de la anterior, pero permitiendo retroceder de forma controlada.
- **Persistencia de Datos:** Guardar todo el progreso del torneo en una base de datos local para que pueda ser retomado en cualquier momento.

3. Arquitectura del Sistema

La aplicación sigue una arquitectura en capas que promueve la separación de responsabilidades, facilitando el mantenimiento y la escalabilidad.

- **Capa de Presentación (UI - gui.py):** Es la parte visual y el controlador principal de la aplicación. Construida con Tkinter, se encarga de renderizar todos los widgets, capturar las interacciones del usuario y orquestar las llamadas a las otras capas.
- **Capa de Lógica de Negocio (logic.py):** Contiene las reglas del torneo. Es un módulo independiente que se encarga de la lógica pura, como la generación de fixtures y el cálculo de enfrentamientos.
- **Capa de Acceso a Datos (database.py):** Actúa como una capa de abstracción sobre la base de datos SQLite. Centraliza todas las operaciones CRUD.

Módulos de Soporte:

- **config.py:** Centraliza todas las constantes y parámetros del torneo.
- **widgets.py:** Contiene componentes de UI personalizados y reutilizables, como la botonera inferior y las filas de partidos.
- **main.py:** Es el punto de entrada que instancia y lanza la aplicación.

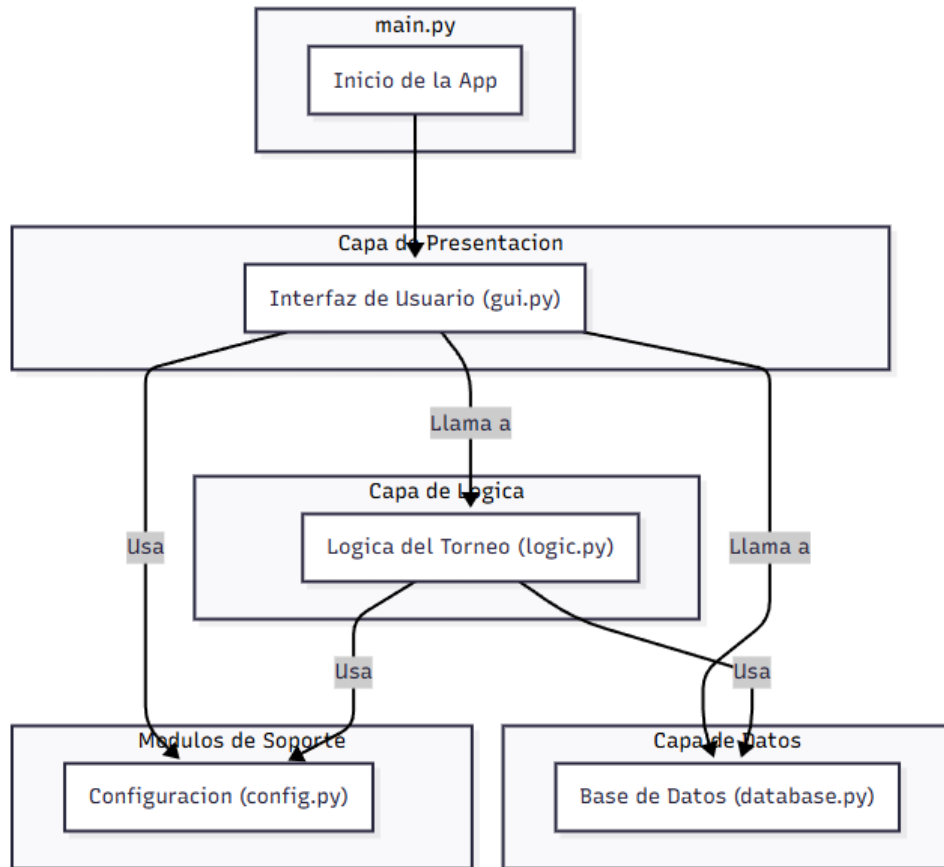


Figura 1: Diagrama de la arquitectura en capas del sistema.

4. Diseño de la Interfaz de Usuario (UI)

La interfaz se organiza en un Notebook (pestañas) para guiar al usuario a través del flujo natural del torneo. Todas las etapas comparten una barra de estado y una botonera inferior con altura fija para una experiencia consistente.

4.1. Etapa 1: Equipos

Propósito: Permitir al usuario definir todos los equipos que participarán en el torneo antes de que comience. Esta etapa se bloquea una vez que todos los cupos están llenos y el usuario lo confirma.


Componentes Clave:

- **Formulario de Entrada:** `ttk.Entry` para el nombre del equipo y `ttk.Combobox` para seleccionar la zona, dispuestos en un panel superior de altura fija.
- **Vista de Lista:** Un `ttk.Treeview` que ocupa el espacio flexible para mostrar la lista de equipos registrados.
- **Botonera Inferior:** Un panel centrado con los botones de acción: “Eliminar Seleccionado”, “Eliminar Datos de Etapa”, “Eliminar Todos los Datos”, “Generar Datos de Prueba” y “Confirmar Etapa”.

4.2. Etapa 2: Grupos

Propósito: Gestionar el desarrollo de la fase de grupos. El usuario selecciona una zona, ve los partidos pendientes y registra los resultados.

Componentes Clave:


Gestor de Torneo de Fútbol

Etapa 1: Equipos

2. Etapa 2: Grupos

Etapa 3: Eliminatorias

Gestión de Equipos

Nombre

Boca Juniors

▼

Zona

A

▼

Agregar

Modificar

Lista de Equipos

ID	Nombre	Zona	Color
145	Equipo-01	A	
146	Equipo-02	A	
147	Equipo-03	B	
148	Equipo-04	B	

Eliminar Seleccionado

Eliminar Datos de Etapa

Eliminar todos los Datos

Generar Datos de Pruebas


Confirmar Etapa

Equipos: 0

Etapa: Equipos

Listo

- **Panel Superior Fijo:** Contiene la lista de partidos pendientes y el formulario de registro de resultados.
- **Tabla de Posiciones:** Un `ttk.Treeview` que ocupa el espacio flexible con las estadísticas de cada equipo.
- **Botonera Inferior:** Panel centrado con los botones: “Volver Etapa”, “Eliminar Datos de Etapa”, “Eliminar Todos los Datos”, “Generar Datos de Prueba” y “Confirmar Etapa”.


Gestor de Torneo de Fútbol

Etapa 1: Equipos
2. Etapa 2: Grupos
Etapa 3: Eliminatórios

Partidos Pendientes

Seleccionar Zona

A

Equipo Local	Equipo Visitante
Equipo-01	Equipo-06
Equipo-01	Equipo-07
Equipo-01	Equipo-08

Registrar Resultado

Seleccione un partido

VS

Guardar

Limpiar

Tabla de Posiciones General

Zona	Equipo	PJ	PG	PE	PP	GF	GC	DG	Puntos
A	Equipo-14	15	9	4	4	42	26	16	31
B	Equipo-31	15	9	3	3	39	25	14	30
B	Equipo-21	15	9	2	4	42	32	10	29

Volver Etapa
Eliminar Datos de Etapa
Eliminar todos los Datos
Generar Datos de Pruebas
Confirmar Etapa

Eiupos: 32
Etapa: Grupos
Etapa 1: Bloqueada
Listo

Figura 3: Wireframe de la Etapa 2: Grupos.

4.3. Etapa 3: Eliminatorias

Propósito: Mostrar y gestionar las rondas de eliminación directa hasta la final.

Componentes Clave:

- **Botón de Generación:** “Generar Octavos de Final” (o la ronda inicial) que aparece cuando es necesario.
- **Vista de Rondas:** Las rondas se generan dinámicamente en un área con scroll. Cada ronda es un `CollapsibleFrame` que contiene widgets `MatchRowWidget`.
- **Vista de Campeón:** Una sección especial que aparece al finalizar el torneo.
- **Botonera Inferior:** Panel centrado con los botones: “Volver Etapa”, “Eliminar Datos de Etapa” y “Eliminar Todos los Datos”.

Figura 4: Wireframe de la Etapa 3: Eliminatorias.

5. Diseño de la Base de Datos

La persistencia de datos se maneja con una base de datos SQLite. El esquema se compone de tres tablas principales:

Tabla equipos: Almacena la información de cada equipo participante.

```
CREATE TABLE IF NOT EXISTS equipos (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    nombre TEXT NOT NULL UNIQUE,  
    zona TEXT NOT NULL,  
    color_hex TEXT NOT NULL UNIQUE  
);
```

Tabla partidos: Almacena cada partido jugado en el torneo. La restricción ‘ON DELETE CASCADE’ asegura que si un equipo es eliminado, todos sus partidos también lo son.

```
CREATE TABLE IF NOT EXISTS partidos (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    fase TEXT NOT NULL,  
    equipo_local_id INTEGER,  
    equipo_visitante_id INTEGER,  
    goles_local INTEGER,
```

```

goles_visitante INTEGER,
ganador_id INTEGER,
FOREIGN KEY (equipo_local_id) REFERENCES equipos (id) ON DELETE CASCADE,
FOREIGN KEY (equipo_visitante_id) REFERENCES equipos (id) ON DELETE CASCADE,
FOREIGN KEY (ganador_id) REFERENCES equipos (id) ON DELETE CASCADE
);

```

Tabla config: Tabla de tipo llave-valor para guardar el estado del torneo (ej: qué etapas están bloqueadas).

```

CREATE TABLE IF NOT EXISTS config (
    llave TEXT PRIMARY KEY,
    valor TEXT NOT NULL
);

```

6. Consideraciones Técnicas y Decisiones de Diseño

- **Framework de UI (Tkinter):** Se eligió Tkinter por ser la librería de GUI estándar de Python, lo que garantiza que la aplicación no tenga dependencias externas pesadas y sea fácilmente distribuable.
- **Base de Datos (SQLite):** Se optó por SQLite para que la aplicación sea totalmente autocontenida. Todos los datos se guardan en un único archivo, facilitando la portabilidad.
- **Manejo de Estado:** El estado del torneo se gestiona a través de la tabla `config`. Esto asegura que la aplicación pueda cerrarse y reabrirse en cualquier momento, continuando donde el usuario la dejó.
- **Separación de Lógica y Presentación:** La decisión más importante fue aislar la lógica de las reglas del torneo en `logic.py`. Esto permite que futuras modificaciones a las reglas no afecten la interfaz gráfica.
- **Componentes Reutilizables:** Widgets complejos como las barras de botones y las filas de partidos se han encapsulado en clases dentro de `widgets.py`. Esto reduce la duplicación de código y simplifica el mantenimiento de la capa de presentación.