⚽ Sistema de Gestión de Torneos de Fútbol

Aplicación de escritorio desarrollada en Python con PySide6 para la gestión completa de torneos de fútbol eliminatorios.

📋 Características Principales

Gestión Completa del Torneo

- **Equipos**: Registro de equipos con información personalizada (nombre, curso, color, escudo)
- **Participantes**: Gestión de jugadores y árbitros con estadísticas detalladas
- **Partidos**: Calendario completo de partidos con resultados y estadísticas
- **Cuadro de Eliminatorias**: Visualización gráfica de octavos, cuartos, semifinales y final
- **Traducciones**: Soporte multiidioma (Español/Inglés)
- **Temas**: Interfaz moderna con diferentes estilos visuales

📊 NUEVA FUNCIONALIDAD: Generación de Informes

La aplicación incluye un potente sistema de generación de informes profesionales en PDF con **tres tipos de reportes**:

1. Informe de Equipos y Jugadores

- Lista completa de equipos con sus plantillas
- Estadísticas individuales de cada jugador (goles, tarjetas)
- Totales por equipo
- Filtrado opcional por equipo específico

2. Informe de Partidos y Resultados

- Calendario completo de partidos
- Resultados y marcadores
- Árbitros asignados
- Estado de los partidos (Pendiente/Jugado)
- Información de penaltis
- Filtrado opcional por fase (octavos, cuartos, etc.)

3. Informe de Clasificación y Eliminatorias

- Tabla de posiciones con estadísticas (PJ, PG, PE, PP, GF, GC, Dif, Pts)
- Ordenamiento automático por puntos y diferencia de goles
- Filtrado opcional por fase eliminatoria

🚀 Instalación y Ejecución

Requisitos

- Python 3.10 o superior
- PySide6
- fpdf2

Instalación

```
# Clonar el repositorio
git clone https://github.com/himlaia/gestion_torneos.git
cd gestion_torneos/torneo_futbol

# Instalar dependencias
pip install -r requirements.txt

# Ejecutar la aplicación
python main.py
```

Primera Ejecución

En el primer inicio, la aplicación creará automáticamente:

- Base de datos SQLite: `data/torneo.db`
- Directorio para escudos: `data/escudos/`
- Directorio para informes generados: `reports/generated/`

Cómo Generar Informes PDF

Desde la Interfaz de Usuario

1. **Abrir la sección de Informes**
    - Navegar a la página "Informes" desde el menú lateral
2. **Seleccionar el tipo de informe**
    - Elegir entre: Equipos y Jugadores, Partidos y Resultados, o Clasificación y Eliminatorias
3. **Aplicar filtros (opcional)**
    - **Equipos y Jugadores**: Seleccionar un equipo específico o "Todos los equipos"
    - **Partidos/Clasificación**: Seleccionar una fase específica o "Todas las fases"
4. **Generar el informe**
    - Click en **"Generar PDF"**: Crea el informe y lo abre automáticamente
    - Click en **"Guardar como..."**: Permite elegir la ubicación de guardado

Ubicación de los Informes

Los PDFs generados automáticamente se guardan en:

```
torneo_futbol/reports/generated/
```

Con nombres que incluyen fecha y hora:

- `equipos_jugadores_20260220_153045.pdf`
- `partidos_resultados_20260220_153126.pdf`
- `clasificacion_eliminatorias_20260220_153201.pdf`

Plantillas JasperReports

La aplicación incluye plantillas profesionales de JasperReports en formato `.jrxml` que pueden ser utilizadas con **JasperSoft Studio** para diseño avanzado de informes.

Ubicación de las Plantillas

```
torneo_futbol/reports/templates/
├── clasificacion_eliminatorias.jrxml
├── partidos_resultados.jrxml
└── equipos_jugadores.jrxml
```

Cómo Usar las Plantillas JasperReports

Opción 1: Edición Visual con JasperSoft Studio

1. **Descargar JasperSoft Studio**
   - Visitar: https://community.jaspersoft.com/downloads
   - Descargar e instalar JasperSoft Studio (versión Community, gratuita)
2. **Abrir las plantillas**
   - Abrir JasperSoft Studio
   - File → Open File
   - Navegar a `reports/templates/` y abrir el archivo `.jrxml` deseado
3. **Configurar la conexión a la base de datos**
   - Crear un Data Adapter para SQLite
   - Apuntar a `data/torneo.db`
   - Driver JDBC: `org.sqlite.JDBC`
   - URL: `jdbc:sqlite:../../../data/torneo.db`
4. **Editar el diseño**
   - Usar el editor visual para modificar campos, estilos, layout
   - Preview para visualizar con datos reales
5. **Exportar el informe**

- Compilar a `.jasper` para usar con JasperReports API
- O exportar directamente a PDF desde JasperSoft Studio

Opción 2: Compilación y Uso con Python (Avanzado)

Para usar las plantillas JasperReports desde Python necesitas:

```
# Instalar JasperReports con Java
pip install JayDeBeApi jpype1

# Descargar JasperReports library (JAR files)
# Disponible en: https://sourceforge.net/projects/jasperreports/
```

Nota: La aplicación actualmente genera PDFs directamente con `fpdf2` sin requerir Java ni JasperReports runtime. Las plantillas `.jrxml` están disponibles para edición personalizada avanzada.

Complicaciones Comunes y Soluciones

❌ Problema: El PDF generado está vacío o sin datos

Causa: La base de datos no tiene información **Solución**:

- Verificar que existen equipos y partidos registrados
- Revisar que los partidos tienen resultados (estado "Jugado")
- Comprobar los filtros aplicados (pueden estar excluyendo todos los datos)

❌ Problema: Error al generar el informe

Causa: Permisos de escritura o carpeta inexistente **Solución**:

```
# Crear manualmente las carpetas necesarias
mkdir -p reports/generated
mkdir -p reports/compiled
chmod 755 reports/generated
```

❌ Problema: No se puede abrir el PDF automáticamente

Causa: Sistema operativo sin visor PDF predeterminado **Solución**:

- El PDF se guarda correctamente en `reports/generated/`
- Abrir manualmente con cualquier visor PDF
- En Windows: Instalar Adobe Reader o usar Microsoft Edge
- En Linux: Instalar `xpdf` o `evince`

❌ Problema: Caracteres con tildes se ven mal en JasperSoft

Causa: Encoding incorrecto **Solución**:

- Verificar que el archivo `.jrxml` está guardado como UTF-8
- En JasperSoft Studio: File → Properties → Text File Encoding: UTF-8
- Las plantillas incluidas ya están corregidas con tildes en formato Unicode

❌ Problema: Fuentes no se cargan correctamente

Causa: Archivo de fuente faltante **Solución**:

- Las fuentes personalizadas están en `app/resources/fonts/`
- Si faltan archivos `.ttf`, la aplicación usa fuentes del sistema automáticamente
- Para usar Poppins: Descargar de Google Fonts y colocar en `resources/fonts/`

Estructura del Proyecto

```
torneo_futbol/
├── main.py                    # Punto de entrada
├── requirements.txt           # Dependencias
├── app/
│   ├── models/              # Modelos de datos (DB)
│   ├── views/               # Interfaz gráfica (PySide6)
│   ├── controllers/         # Lógica de negocio
│   ├── services/            # Servicios (informes, estilos, eventos)
│   ├── resources/           # Imágenes, fuentes, estilos CSS
│   └── config.py            # Configuración global
├── data/                    # Base de datos y escudos
├── reports/
│   ├── templates/           # Plantillas JasperReports (.jrxml)
│   ├── generated/           # PDFs generados
│   └── compiled/            # Archivos .jasper compilados
├── translations/            # Archivos de traducción (.ts, .qm)
└── docs/                    # Documentación adicional
```

Tecnologías Utilizadas

- **PySide6**: Framework Qt para interfaces gráficas
- **SQLite**: Base de datos ligera y sin servidor
- **fpdf2**: Generación de PDFs profesionales
- **JasperReports**: Diseño avanzado de plantillas de informes (opcional)

Arquitectura

El proyecto sigue el patrón **MVC (Model-View-Controller)**:

- **Models** (`app/models/`): Acceso a datos y operaciones de base de datos
- **Views** (`app/views/`): Componentes visuales de la interfaz
- **Controllers** (`app/controllers/`): Lógica de negocio y coordinación entre modelos y vistas
- **Services** (`app/services/`): Servicios transversales (informes, estilos, eventos)

Construcción del Ejecutable

Para generar un archivo `.exe` standalone:

```
# Instalar PyInstaller
pip install pyinstaller

# Ejecutar script de construcción
cd scripts
./build.ps1
```

El ejecutable se generará en `build/torneo_futbol/TorneoFutbol.exe`

📖 Documentación Adicional

- **Guía de Usuario**: `docs/GUIA_USUARIO.md`
- **Scripts de Utilidad**: `scripts/README.md`
- **Detalles de Entrega**: `README_ENTREGA.md`

🤝 Contribución

Este proyecto fue desarrollado como parte de un trabajo académico.

📜 Licencia

Proyecto académico - Uso educativo

Desarrollado con ❤ usando Python y PySide6