# MiFutbolC - Proyecto De Uso Personal

## Descripción

MiFutbolC es un sistema de gestión de datos para fútbol desarrollado en lenguaje C. Permite administrar camisetas, canchas, partidos, estadísticas, logros, análisis de rendimiento, lesiones y exportar datos en múltiples formatos. Utiliza SQLite como base de datos para almacenar toda la información de manera persistente y eficiente.

El proyecto está diseñado para ser una herramienta completa para el seguimiento y análisis de datos relacionados con el fútbol, desde la gestión de equipamiento hasta el registro de partidos, canchas y lesiones.

## Características Principales

- **Gestión de Camisetas**: Crear, listar, editar y eliminar camisetas de fútbol.
- **Gestión de Canchas**: Gestionar canchas de fútbol.
- **Gestión de Partidos**: Registrar partidos con detalles como cancha, goles, asistencias y camiseta utilizada.
- **Estadísticas**: Visualizar estadísticas agregadas del sistema.
- **Análisis de Rendimiento**: Comparar el rendimiento de los últimos 5 partidos con promedios generales y calcular rachas de victorias y derrotas.
- **Gestión de Lesiones**: Registrar y gestionar lesiones de jugadores.
- **Importación de Datos**: Importar datos desde archivos JSON a la base de datos.
- **Exportación de Datos**: Exportar todos los datos en formatos CSV, TXT, JSON y HTML.
- **Interfaz de Menú**: Navegación intuitiva mediante menús interactivos.
- **Base de Datos SQLite**: Almacenamiento persistente y eficiente de datos.
- **Documentación Doxygen**: Generación automática de documentación técnica.

## Requisitos del Sistema

- **Compilador C**: Compatible con GCC o MinGW (incluido en CodeBlocks).
- **SQLite3**: Biblioteca incluida en el proyecto (sqlite3.c y sqlite3.h).
- **cJSON**: Biblioteca incluida en el proyecto (cJSON.c y cJSON.h), bajo licencia MIT.
- **Sistema Operativo**: Windows, Linux o macOS.
- **Herramientas de Desarrollo**:
  - CodeBlocks (recomendado para compilar el proyecto .cbp).
  - Doxygen (opcional, para generar documentación).

## Instalación y Compilación

### Opción 1: Usando CodeBlocks (Recomendado)

1. Instala CodeBlocks desde codeblocks.org.
2. Abre el archivo `MiFutbolC.cbp` con CodeBlocks.
3. Compila el proyecto seleccionando “Build” > “Build”.
4. El ejecutable se generará en `bin/Debug/MiFutbolC.exe`.

### Opción 2: Compilación Manual con GCC

1. Asegúrate de tener GCC instalado.
2. Navega al directorio raíz del proyecto.
3. Compila todos los archivos fuente:

```
gcc -o MiFutbolC main.c db.c menu.c camiseta.c partido.c estadisticas.c analisis.c cancha.c logros.c lesi
```

4. Ejecuta el programa:

```
./MiFutbolC
```

## Uso

Al ejecutar el programa, se presenta un menú principal con las siguientes opciones:

1. **Camisetas**: Gestionar camisetas (crear, listar, editar, eliminar).
2. **Canchas**: Gestionar canchas de fútbol.
3. **Partidos**: Gestionar partidos (crear, listar, modificar, eliminar).
4. **Estadísticas**: Mostrar estadísticas generales del sistema.
5. **Logros**: Gestionar logros y badges.
6. **Análisis**: Mostrar análisis de rendimiento.
7. **Lesiones**: Gestionar lesiones de jugadores.
8. **Exportar Todo**: Exportar todos los datos en múltiples formatos.
9. **Importar Todo**: Importar todos los datos desde archivos JSON.
10. **Salir**: Cerrar el programa.

#### Ejemplo de Uso

- Selecciona “1” para acceder al menú de camisetas.
- Elige “1” para crear una nueva camiseta.
- Ingresa el nombre cuando se solicite.
- La camiseta se guardará en la base de datos.

## Estructura del Proyecto

```
MiFutbolC/
├── main.c                # Punto de entrada del programa
├── db.c / db.h           # Gestión de la base de datos SQLite
├── menu.c / menu.h       # Sistema de menús interactivos
├── menu_camisetas.h      # Declaraciones específicas para menús de camisetas
├── models.h              # Definiciones de estructuras comunes
├── camiseta.c / camiseta.h # Gestión de camisetas
├── partido.c / partido.h   # Gestión de partidos
├── estadisticas.c / estadisticas.h # Cálculo y visualización de estadísticas
├── analisis.c / analisis.h # Análisis de rendimiento
├── cancha.c / cancha.h    # Gestión de canchas
├── logros.c / logros.h    # Gestión de logros
├── lesion.c / lesion.h    # Gestión de lesiones
├── export.c / export.h     # Funciones de exportación individuales
├── export_all.c / export_all.h # Exportación completa de datos
├── import.c / import.h     # Funciones de importación desde JSON
├── utils.c / utils.h       # Utilidades auxiliares
├── cjson/                # Directorio de la biblioteca cJSON
│   ├── cJSON.c / cJSON.h       # Biblioteca cJSON para manejo de JSON
│   └── cJSON_Utils.c / cJSON_Utils.h # Utilidades adicionales para cJSON
├── sqlite3.c / sqlite3.h   # Biblioteca SQLite embebida
├── MiFutbolC.cbp          # Proyecto CodeBlocks
├── data/                 # Directorio de datos
│   ├── mifutbol.db        # Base de datos SQLite
│   ├── *.csv              # Archivos exportados en CSV
│   ├── *.txt              # Archivos exportados en TXT
│   ├── *.json             # Archivos exportados en JSON
│   └── *.html             # Archivos exportados en HTML
├── bin/                  # Binarios compilados
│   └── Debug/
│       └── MiFutbolC.exe
```

```
├── doxygen/              # Documentación generada
│   ├── doxyfile          # Configuración de Doxygen
│   └── html/             # Documentación HTML
└── obj/                  # Archivos objeto de compilación
```

## Base de Datos

El proyecto utiliza SQLite para almacenar datos. La base de datos se crea automáticamente en `data/mifutbol.db` al ejecutar el programa por primera vez.

#### Tablas Principales

- **camiseta**: Almacena información de camisetas (ID, nombre, sorteada).
- **cancha**: Gestiona canchas de fútbol (ID, nombre, ubicacion).
- **partido**: Registra partidos (ID, cancha_id, fecha/hora, goles, asistencias, rendimiento, cansancio, animo, camiseta_id).
- **lesion**: Gestiona lesiones (ID, jugador, tipo, fecha, duracion).
- **logros**: Almacena logros y badges (ID, nombre, descripcion, nivel, objetivo, categoria).
- **estadisticas**: Contiene estadísticas calculadas (ID, tipo, valor, camiseta_id).

#### Inicialización

La función `db_init()` en `db.c` se encarga de: - Abrir la conexión a la base de datos. - Crear las tablas si no existen. - Preparar el esquema de datos.

## Funcionalidades de Exportación

El sistema permite exportar datos en múltiples formatos:

- **CSV**: Formato de valores separados por comas, ideal para hojas de cálculo.
- **TXT**: Texto plano formateado para lectura humana.
- **JSON**: Formato estructurado para integración con otras aplicaciones.
- **HTML**: Páginas web con tablas para visualización en navegador.

Los archivos exportados se guardan en el directorio `data/` con nombres descriptivos como `camisetas.csv`, `partidos.html`, etc.

## Módulo de Estadísticas

El módulo de estadísticas proporciona información agregada sobre el rendimiento de las camisetas en los partidos:

- **Camiseta con más Goles**: Muestra la camiseta que ha acumulado el mayor número de goles en todos los partidos.
- **Camiseta con más Asistencias**: Identifica la camiseta con el mayor número de asistencias registradas.
- **Camiseta con más Partidos**: Lista la camiseta que ha sido utilizada en el mayor número de partidos.
- **Camiseta con más Goles + Asistencias**: Combina goles y asistencias para determinar la camiseta con mejor rendimiento global.

Las estadísticas se calculan en tiempo real mediante consultas SQL que unen las tablas de partidos y camisetas.

## Módulo de Análisis de Rendimiento

El módulo de análisis de rendimiento (`analisis.c`) ofrece una evaluación detallada del desempeño futbolístico mediante la comparación de los últimos 5 partidos con los promedios generales del sistema:

- **Comparación Últimos 5 vs Promedio General**: Analiza métricas como goles, asistencias, rendimiento general, cansancio y estado de ánimo, mostrando diferencias numéricas entre el rendimiento reciente y el histórico.

- **Cálculo de Rachas**: Determina la mejor racha de victorias consecutivas y la peor racha de derrotas consecutivas registradas.
- **Análisis Motivacional**: Proporciona mensajes personalizados basados en el rendimiento comparativo, ofreciendo motivación o consejos constructivos para mejorar.
- **Visualización de Últimos Partidos**: Muestra un resumen de los 5 partidos más recientes con detalles clave como fecha, goles, asistencias, rendimiento y resultado.

Este módulo utiliza consultas SQL avanzadas para calcular promedios y rachas, proporcionando insights valiosos para el seguimiento y mejora del rendimiento futbolístico.

## Sistema de Logros y Badges

El sistema de logros y badges (`logros.c`) implementa un sistema de recompensas basado en estadísticas conseguidas por las camisetas en partidos de fútbol, incentivando el progreso y el logro de metas:

- **Categorías de Logros**: Incluye logros por goles, asistencias, partidos jugados, contribuciones totales (goles + asistencias), victorias, empates, derrotas, rendimiento general, estado de ánimo, canchas distintas, hat-tricks, poker de asistencias, rendimiento perfecto, ánimo perfecto, y logros específicos en victorias, derrotas y empates.
- **Niveles de Dificultad**: Cada categoría tiene múltiples niveles (Novato, Promedio, Experto, Maestro, Leyenda) con objetivos progresivos.
- **Seguimiento de Progreso**: Muestra el progreso actual hacia cada logro, indicando si está no iniciado, en progreso o completado.
- **Visualización por Camiseta**: Permite ver todos los logros, solo los completados o solo los en progreso para una camiseta específica.
- **Interfaz de Menú**: Navegación intuitiva para explorar los logros disponibles.

Este sistema utiliza consultas SQL para calcular estadísticas acumuladas y determinar el estado de cada logro, proporcionando una experiencia gamificada para motivar el uso continuo del sistema.

## Utilidades y Funciones Auxiliares

El proyecto incluye un módulo de utilidades (`utils.c / utils.h`) que proporciona funciones comunes para:

- **Entrada de Datos**: `input_int()` y `input_string()` para leer enteros y cadenas del usuario con validación básica.
- **Manejo de Fecha/Hora**: `get_datetime()` para obtener fecha y hora actual en formato legible, y `get_timestamp()` para nombres de archivos.
- **Interfaz de Consola**: `clear_screen()`, `print_header()`, y `pause_console()` para mejorar la experiencia del usuario en terminal.
- **Validación de Datos**: `existe_id()` para verificar si un ID existe en una tabla de la base de datos.
- **Confirmaciones**: `confirmar()` para solicitar confirmación del usuario antes de operaciones destructivas.
- **Gestión de Exportaciones**: `obtener_directorio_exports()` para determinar y crear el directorio de exportación (en Escritorio en Windows, home en Unix/Linux).

Estas utilidades promueven la reutilización de código y mantienen una interfaz consistente en todo el programa.

## Arquitectura y Diseño

#### Patrón de Diseño Modular

El proyecto sigue un enfoque modular con separación clara de responsabilidades:

- **Separación de Interfaz y Lógica**: Cada módulo tiene archivos `.h` (interfaces/declaraciones) y `.c` (implementaciones).
- **Bajo Acoplamiento**: Los módulos interactúan principalmente a través de funciones públicas, minimizando dependencias directas.
- **Alto Cohesión**: Cada módulo se enfoca en una responsabilidad específica (ej. `camiseta.c` solo maneja camisetas).

**Patrón de Diseño de Menús**

El sistema de menús implementa un patrón de Comando simplificado:

- **Estructura MenuItem**: Define comandos con número de opción, texto descriptivo y función a ejecutar.
- **Función ejecutar_menu()**: Actúa como invocador que muestra opciones y ejecuta comandos seleccionados.
- **Jerarquía de Menús**: Menú principal con submenús especializados, permitiendo navegación intuitiva.

**Patrón de Acceso a Datos**

- **Capa de Abstracción de BD**: `db.c/db.h` centraliza todas las operaciones de base de datos.
- **SQL Embebido**: Consultas SQL directamente en el código C para máxima flexibilidad.
- **Gestión de Conexión**: Conexión única mantenida durante toda la ejecución del programa.

## Sistema de Menús

El proyecto implementa un sistema de menús jerárquico y modular mediante las funciones en `menu.c / menu.h`:

- **Menú Principal**: Gestionado en `main.c`, presenta las opciones principales del sistema (Camisetas, Canchas, Partidos, Estadísticas, Logros, Análisis, Lesiones, Exportar Todo, Importar Todo, Salir).
- **Submenús**: Cada módulo principal tiene su propio menú (ej. `menu_camisetas()`, `menu_canchas()`, `menu_partidos()`, `menu_logros()`, `menu_lesiones()`).
- **Estructura de Menú**: Utiliza la estructura `MenuItem` definida en `models.h` para asociar opciones numéricas con textos descriptivos y funciones a ejecutar.
- **Navegación**: La función `ejecutar_menu()` maneja la lógica de mostrar opciones, leer selección del usuario y ejecutar la acción correspondiente.
- **Consistencia**: Todos los menús siguen el mismo patrón, facilitando la adición de nuevas funcionalidades.

## Exportación Completa

Además de las exportaciones individuales, el módulo `export_all.c / export_all.h` proporciona la función `exportar_todo()` que:

- Ejecuta automáticamente todas las funciones de exportación disponibles.
- Genera archivos en formatos CSV, TXT, JSON y HTML para camisetas, partidos, estadísticas y lesiones.
- Facilita la copia de seguridad completa de todos los datos del sistema.
- Es accesible directamente desde el menú principal como opción “Exportar Todo”.

## Importación Completa

Además de las exportaciones, el módulo `import.c / import.h` proporciona la función `importar_todo()` que:

- Permite importar datos desde archivos JSON generados por la función de exportación.
- Valida la estructura de los datos JSON antes de insertarlos en la base de datos.
- Maneja errores de importación y proporciona feedback al usuario.
- Facilita la restauración de datos desde copias de seguridad.
- Es accesible directamente desde el menú principal como opción “Importar Todo”.

## Documentación

La documentación técnica se genera automáticamente usando Doxygen:

1. Instala Doxygen.
2. Ejecuta doxygen `doxygen/doxyfile` desde el directorio raíz.
3. Abre `doxygen/html/index.html` en un navegador web.

## Desarrollo y Contribución

### Convenciones de Código

- El código sigue estándares de C para legibilidad y mantenibilidad.
- Comentarios en español para consistencia.
- Uso de headers (.h) para declaraciones y archivos .c para implementaciones.

### Agregar Nuevas Funcionalidades

1. Define la funcionalidad en el archivo .h correspondiente.
2. Implementa en el archivo .c.
3. Actualiza el menú principal si es necesario.
4. Agrega opciones de exportación si aplica.

### Pruebas

- Compila y ejecuta el programa después de cambios.
- Verifica la integridad de la base de datos.
- Prueba todas las opciones de menú.

## Licencia

Este proyecto es de código abierto. Consulta el archivo LICENSE si está presente.

## Autor

Proyecto desarrollado por Thomas Hamer como ejemplo educativo y de uso personal de programación en C con SQLite.

## Notas Adicionales

- Los datos se persisten entre ejecuciones gracias a SQLite.
- El programa maneja errores básicos y solicita confirmaciones para operaciones destructivas.
- Compatible con Windows (probado en Windows 11).
- La interfaz es completamente textual, no requiere GUI.

## Manual de Usuario

Para una guía detallada de uso del programa, incluyendo instrucciones paso a paso para cada funcionalidad, ejemplos de uso y solución de problemas comunes, consulta el Manual de Usuario.

Este manual incluye capturas de pantalla de los menús y secciones del programa para facilitar la navegación.

Para más información, consulta la documentación generada con Doxygen o revisa el código fuente comentado.