Sistema de Gestión de Torneos eSports 'Nexus Arena'

Contexto General

Bienvenidos a **"Nexus Arena"**, una organización en crecimiento dedicada a la gestión, promoción y realización de torneos profesionales y *amateur* de *eSports*. Actualmente, la inscripción y el seguimiento de los eventos se realiza de manera caótica a través de múltiples *spreadsheets* y formularios, causando confusión y errores en las *brackets*.

Su equipo de desarrollo tiene la misión de crear un **Sistema de Gestión de Torneos (SGT)** moderno y eficiente, desarrollado íntegramente en **lenguaje C**. Este SGT debe centralizar la información de los **usuarios** (jugadores), los **videojuegos** (*games*) y el estado de los **torneos**, garantizando la persistencia de los datos para ofrecer una experiencia fluida a los competidores y facilitar el trabajo del personal de la organización.

Estructuras de Datos Base

El sistema debe utilizar las siguientes estructuras base para modelar la operación de la organización. Las mismas son demostrativas y podrán ser modificadas por los grupos de ser requerido.

```c
// Estructuras de Juegos, Usuarios, Administradores, Fechas y Torneos
typedef struct stVideojuego {
    char idJuego[10]; // Código único del juego (ej. "LOL", "CS2")
    char nombre[50];
    char genero[30];
    char plataforma[20];
} Videojuego;

typedef struct stUsuario {
    char idUsuario[10]; // ID de jugador único
    char nickname[30];
    char email[50];
    char pais[30];
    int nivel; // Nivel de experiencia (ej. 1 a 10)
} Usuario;

typedef struct stAdministrador {
    char usuario[20]; // Usuario para el login
    char contrasenia[20];
    char nombreCompleto[50];
    char rol[20]; // Ej: "Organizador", "Staff"
} Administrador;
```

```
typedef struct stFecha {
    int dia;
    int mes;
    int anio;
} Fecha;

typedef struct stTorneo {
    char idTorneo[10];
    char nombre[50];
    Videojuego juego;
    int capacidadMaxima;
    int cuposDisponibles;
    Fecha fechaInicio;
    float premioTotal;
    char estado[15]; // "Abierto", "En Curso", "Finalizado"
} Torneo;

typedef struct stInscripcion {
    char idInscripcion[10];
    char idTorneo[10];
    char idUsuario[10];
    Fecha fechaInscripcion;
} Inscripcion;
```

Requisitos Mínimos e Indispensables

El SGT de "Nexus Arena" debe garantizar la **seguridad y la persistencia de los datos**. Los datos de **usuarios**, **torneos**, **videojuegos** e **inscripciones** deben almacenarse en **archivos binarios** separados.

I. Acceso al Sistema

1. **Login y Seguridad:** El sistema debe iniciar con un **módulo de *login*** para el personal administrativo. El administrador deberá ingresar credenciales (`usuario` y `contrasenia`) que serán validadas contra una lista de `Administradores` predefinida.
2. **Menú Principal:** Una vez autenticado, el administrador accederá a un menú de opciones para gestionar la organización.

II. Gestión de Entidades (CRUD Básico)

El sistema debe permitir gestionar el catálogo de juegos, los jugadores y los torneos.

1. **Gestión de Torneos:**

    - **3a. Crear / Modificar Torneo:** Permite crear un nuevo evento (debe estar asociado a un `idJuego` ya existente) o modificar los detalles de uno existente, como el estado o el premio.
2. **Gestión de Usuarios:**
    - **3b. Agregar / Modificar Usuario:** Permite registrar un nuevo jugador en la plataforma o actualizar sus datos (ej. *nickname*).
    - **3c. Ver Listado de Usuarios:** Mostrar un resumen (*nickname*, ID de Usuario) de todos los jugadores registrados.
    - **3d. Ver Info Detallada de Usuario:** Buscar un usuario por ID y mostrar todos sus datos.
3. **Gestión de Videojuegos:**
    - **3e. Agregar / Modificar Videojuego:** Permite listar nuevos títulos de *eSports* soportados.
    - **3f. Ver Catálogo de Videojuegos:** Mostrar la lista de todos los juegos soportados (*idJuego*, nombre, plataforma).

III. Gestión de Inscripciones y Reportes

La gestión de la participación es el núcleo del SGT.

1. **3g. Ver Listado de Torneos:** Mostrar un resumen de todos los torneos (nombre, juego, estado).
2. **3h. Ver Torneos Abiertos (para inscripción):** Mostrar los torneos cuyo `estado` es "Abierto" y cuyo `cuposDisponibles` es mayor que cero.
3. **3i. Ver Inscripciones Resumen:** Mostrar un listado de todas las inscripciones (ID de Torneo, ID de Usuario).
4. **3j. Registrar una Inscripción:**
    - El administrador busca el torneo por ID y el usuario por ID.
    - **Validación:** El torneo debe estar "Abierto" y `cuposDisponibles` debe ser mayor que cero.
    - Si es válido, crea un nuevo registro de `Inscripcion` y actualiza el archivo de torneos: **Decrementa el `cuposDisponibles` del torneo en 1.**
5. **3k. Ver Participantes de un Torneo:** Buscar un torneo por ID y mostrar el listado de todos los `Usuarios` inscritos en él.
6. **3l. Reporte de Torneos Llenos:** Listar todos los torneos cuyo `cuposDisponibles` sea igual a cero y cuyo estado sea "Abierto" o "En Curso".
7. **3m. Reporte de Actividad por Juego:** Calcular cuántos torneos se realizaron para un `idJuego` específico en un año determinado.

Requisito Adicional para Promoción

Para que el grupo de desarrollo pueda optar por la **promoción** de la materia, deberán proponer e implementar una **funcionalidad clave extra por cada miembro del grupo**, que vaya más allá de los requisitos mínimos y mejore significativamente la operación o el análisis en "Rueda Veloz". Estas funcionalidades deberán ser propuestas por el equipo y deberán ser el resultado de un análisis crítico de la problemática.

Nota: La funcionalidad debe ser **relevante**, **compleja** (requerir el uso integrado de archivos y estructuras) y debe ser aprobada previamente por el docente. La documentación de esta funcionalidad y su código serán clave en la evaluación de la promoción.