**Manual Técnico - Sistema de Gestión de Personajes**

**Estructura de la Aplicación**

El sistema está compuesto por dos clases principales:

* Practica1Repositorio: Clase principal que contiene el menú de gestión y la interacción con el usuario.
* GestorPersonajes1: Clase que maneja toda la lógica de operaciones con los personajes.

Arquitectura general

* Capa de Presentación: Maneja la interacción con el usuario (menús, entradas, salidas)
* Capa de Lógica: Gestiona todas las operaciones con los datos de personajes
* Capa de Datos: Almacena la información en arreglos estáticos

**Descripción de Métodos**

Clase Practica1Repositorio

**MenuGestionesPersonajes()**

Propósito: Mostrar el menú principal y gestionar las opciones seleccionadas por el usuario.

Flujo:

* Muestra un menú con 9 opciones
* Captura la selección del usuario
* Valida que la entrada sea texto (para nombres)
* Redirige a los métodos correspondientes en GestorPersonajes1
* Repite hasta que el usuario seleccione salir (opción 9)

**Clase GestorPersonajes1**

Variables estáticas

* historial\_peleas: Matriz para almacenar historial de peleas
* idpersonajes, personajesNombre, armas, niveles: Arreglos para datos básicos
* habilidades: Matriz para habilidades de cada personaje
* Contadores: contadorId, posicionesPersonajes, contadorHistorial

**AgregarPesonajes(String nombre\_personaje, String arma)**

Propósito: Agregar un nuevo personaje al sistema.

Validaciones:

* Verifica que no haya espacios en blanco
* Comprueba que el nombre no exista ya
* Valida que el nombre no contenga números

Proceso:

1. Asigna ID automático
2. Almacena nombre, arma y nivel aleatorio (0-100)
3. Solicita 5 habilidades (valida que no se repitan)

**habilidadRepetida(String nhabilidad, int indice\_recorrido, int pos)**

Propósito: Verificar si una habilidad ya existe para un personaje.

Retorno: true si la habilidad ya existe, false si no.

**ModificarPersonajes(String nam)**

Propósito: Modificar datos de un personaje existente.

Opciones:

1. Cambiar arma
2. Modificar habilidades (por posición)
3. Cambiar nivel de poder
4. Validaciones: Verifica que exista el personaje.

**eliminarPersonajes(String borrar\_personaje)**

Propósito: Eliminar un personaje del sistema.

Proceso:

1. Busca el personaje
2. Elimina todos sus datos
3. Reorganiza los arreglos para evitar huecos
4. Ajusta contadores

**verDatosDeUnPersonaje(String name)**

Propósito: Mostrar toda la información de un personaje específico.

Muestra: ID, nombre, arma, nivel y todas sus habilidades.

**verListadoPersonajes()**

Propósito: Mostrar todos los personajes registrados en el sistema.

Formato: Lista detallada con todos los datos de cada personaje.

**peleaPersoanejes(String personaje1, String personaje2)**

Propósito: Simular una pelea entre dos personajes.

Mecánica:

* Compara los niveles de poder
* El personaje con mayor nivel gana
* Si son iguales, es empate
* Registro: Guarda los resultados en el historial con fecha/hora.

**historialPeleas()**

Propósito: Mostrar todas las peleas registradas.

Información: Fecha, participantes, ganador y perdedor.

**creditos()**

Propósito: Mostrar información del desarrollador.

Contenido: Nombre, carné, curso y universidad.

Lógica General del Sistema

Almacenamiento: Todos los datos se guardan en arreglos estáticos.

Validaciones:

* Entradas de texto no pueden contener números
* No se permiten espacios en blanco
* Habilidades no pueden repetirse
* Identificadores: Se generan automáticamente (ID1, ID2, etc.)
* Niveles: Se asignan aleatoriamente al crear un personaje (0-100)
* Persistencia: Los datos permanecen mientras la aplicación esté en ejecución.

**Flujo Principal**

1. El usuario interactúa con el menú principal
2. Según la opción seleccionada:

Se solicitan datos adicionales si es necesario

Se validan las entradas

Se ejecuta la operación correspondiente

Se muestran resultados o confirmaciones

1. El ciclo continúa hasta que el usuario elija salir

**Consideraciones Técnicas**

* Manejo de arreglos: Se usan contadores para saber cuántas posiciones están ocupadas
* Eliminación: Los elementos eliminados provocan un reacomodo de los arreglos
* Historial: Las peleas se registran con fecha/hora exacta
* Interfaz: Se usan formatos visuales para mejorar la presentación