Manual Técnico Arena Usac

El sistema fue desarrollado en Java utilizando la arquitectura MVC (Modelo-Vista-Controlador), lo que permite que esté organizado, sea escalable y fácil de mantener, está diseñado para que se puedan realizar cambios o mejoras sin complicaciones.

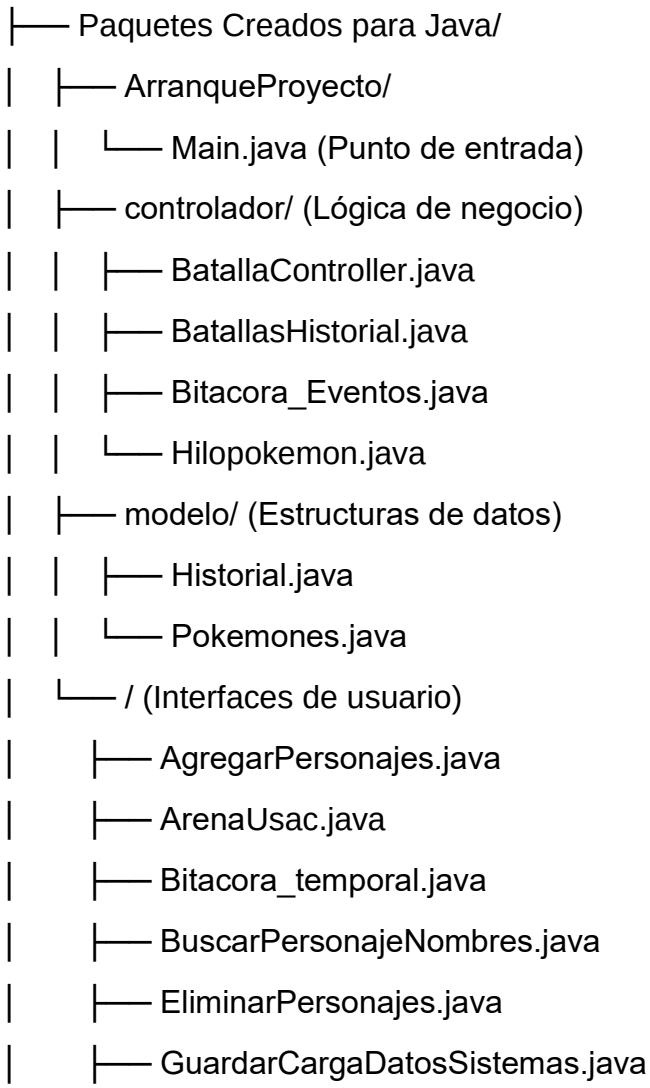
El programa está dividido en 4 carpetas principales:

ArranqueProyecto - Donde inicia todo (el Main.java)

controlador - Los "cerebros" que hacen que todo funcione.

modelo - Donde se guardan los datos de pokemones y batallas.

vista - Todas las pantallas que ves en el programa.



COMPONENTES PRINCIPALES

1.1. Capa de Modelo (modelo)

- Pokemones.java: Clase principal que representa los personajes/pokemones

que guarda sus atributos y su constructor.

- Historial.java: Maneja el registro histórico de batallas, teniendo su propio constructor para guardar datos que mas tarde se mostraran.

1.2. Capa de Controlador (controlador)

- BatallaController.java: Controla la lógica de las validaciones de lo pokemones, registros, existencia y eliminación de cada uno de ellos, no es que controle batallas si no que controla acciones que se puede utilizar para los pokemones a través de un vector de Objetos [ ].

- BatallasHistorial.java: Gestiona el historial de combates

- Bitacora_Eventos.java: Registra eventos del sistema

- Hilopokemon.java: Maneja la el comportamiento de los pokemones entre ellos y hace subprocesos para el funcionamiento correcto de cada batalla que tenga los pokemones.

1.3. Capa de Vista (vista)

Interfaces gráficas para cada funcionalidad:

AgregarPersonajes.java: Interfaz para crear nuevos personajes/pokemones

- Campos para todos los atributos del personaje (nombre, HP, ataque, etc.)
- Validación en tiempo real de rangos numéricos
- Confirmación de registro exitoso
- Prevención de duplicados por nombre/ID

ModificarPersonajes.java: Permite actualizar información de personajes existentes

- Búsqueda de personaje por ID o nombre
- Visualización de datos actuales antes de modificar
- Edición selectiva de atributos (no requiere modificar todos)
- Validación de nuevos valores antes de guardar

EliminarPersonajes.java: Eliminación de registros y remueve personajes del sistema de manera segura

- Búsqueda y visualización previa a la eliminación
- Mecanismo de confirmación.
- Verificación de que el personaje no esté en batallas activas
- Mensaje de eliminación exitosa

VisualizarPersonajesRegistrados.java: Muestra tabla completa de todos los personajes.

- Vista en formato tabla con todos los atributos
- Ordenamiento por diferentes columnas
- Capacidad de actualización en tiempo real
- Navegación entre registros si son muchos

BuscarPersonajeNombres.java: Búsqueda específica

- Búsqueda por nombre parcial o completo
- Visualización detallada de atributos individuales
- Estadísticas de batallas (ganadas/perdidas)
- Acceso rápido al historial específico del personaje

SimularBatalia.java: Simulador de combates

- Selección de dos combatientes por ID o nombre
- Visualización de atributos comparativos
- Bitácora en tiempo real del desarrollo del combate
- Resultado final con detalles del ganador
- Opción de guardar la batalla en historial

HistorialBatalias.java: Consulta de historial

- Tabla con todas las batallas simuladas
- Registros por fecha, participantes o ganador
- Capacidad de ver bitácoras completas de batallas específicas
- Estadísticas generales del sistema

GuardarCargaDatosSistemas.java: Persistencia de datos

- Carga inicial de datos al iniciar la aplicación
- Guardado automático de todos los elementos.
- Exportación manual de datos a archivos de texto
- Importación desde archivos externos
- Validación de integridad de datos

VerdatosEstudiante.java: Información del desarrollador

- Información del desarrollador (nombre, carné, curso)
- Datos de la universidad y repositorio del proyecto

FUNCIONALIDADES TÉCNICAS

2.1. Gestión de Personajes

Creación: Validación de rangos (HP: 100-500, Ataque: 10-100, etc.)

Modificación: Actualización parcial/complete de atributos

Eliminación: Remoción segura con confirmación

Búsqueda: Por ID o nombre con resultados en tiempo real

2.2. Sistema de Batallas

Simulación: Utilización de hilos basado en atributos de personajes

Bitácora: Registro detallado de eventos del combate

Historial: Almacenamiento permanente de resultados

2.3. Gestión de Datos

Carga inicial: Lectura de archivos al iniciar sistema (pendiente)

Guardado automático: Persistencia de datos al utilizar el programa.

Exportación: Generación de reportes en formato texto

2.4. Comunicación entre Componentes

Vista Controlador: Envío de datos de formularios

Controlador Modelo: Lógica dentro del programa que conecta el controlador y la vista donde valida todos los datos necesarios.

Modelo Vista: Actualización de interfaces mostrando resultados de búsquedas y presenta resultados de búsquedas.

Características Técnicas de Hilos.

3.1. Manejo de Hilos (Hilopokemon.java)

- Subprocesos independientes para cada batalla
- Sincronización para evitar conflictos
- Manejo de tiempo real durante combates
- Finalización segura de procesos

3.2. Validaciones en Tiempo Real

- Verificación inmediata de rangos numéricos
- Prevención de duplicados durante escritura
- Validación de formatos antes de procesar
- Mensajes de error específicos y útiles

3.3. Gestión de almacenamiento y persistencia de Datos (Vectores)

- Vectores estáticos que tiene como limite 100 Pokemones.
- Liberación de recursos cuando ya no se usan (Eliminación)
- Optimización para muchos registros (Vector de objetos)
- Manejo eficiente de objetos en memoria

Manejo de Problemas Técnicos

Solución de Incidencias Comunes

- Problemas de rendimiento: Revisar la gestión de hilos en Hilopokemon.java
- Errores de datos: Verificar las validaciones en BatallaController.java
- Fallos de interfaz: Comprobar la comunicación entre vista y controlador

Mejoramiento del Sistema

- Gestión de memoria: Monitorear el uso de vectores con grandes volúmenes de datos (Memoria Dinámica).
- Eficiencia en búsquedas: Mejorar algoritmos para consultas más rápidas

Consideraciones Técnicas Para el Uso del Programas.

Estructura del Proyecto

Mantener la organización actual de paquetes y clases

Preservar la separación clara entre modelo, vista y controlador

Conservar los nombres y ubicaciones de archivos existentes

Estructuraciones de Datos

Validaciones: Los rangos numéricos deben mantenerse consistentes

Persistencia: La estructura de archivos .txt no debe modificarse

Hilos: La lógica de concurrencia requiere especial atención