Informe de aplicacion Proyecto Acceso a Datos Egoitz Perez de Arrilucea 3 Grado Superior Desarrollo de Aplicaciones Multiplataforma 2022-2023

Descripcion:

La aplicacion es un juego de combate mediante consola.
El usuario puede crear su propio jugador y luchar contra diferentes enemigos, con cada combate se sube de nivel que posteriormente permitira mejorar caracteristicas.
El jugador y la partida pueden ser guardadas en una base de datos eXist, el jugador se puede volver a cargar en otra partida y tambien se pueden modificar sus atributos.

BBDD:

El gestor de base de datos de este proyecto es eXist, la base de datos se llama “proyecto” contiene dos colecciones xml: “Jugadores.xml” y “partidas.xml”.

1. Jugadores xml:
    Esta colleccion tiene la siguiente estructura:
    1. jugadores
        1. jugador
            1. nombre
            2. vida (@mejoras)
                1. puntosVida
            3. ataque(@mejoras)
                1. puntosAtaque
            4. nivel

2. partidas.xml:
    Esta colleccion tiene la siguiente estructura:
    1. partidas
        1. fecha
        2. puntuacionTotal
        3. combates
            1. combate (@numero, @resultado)
                1. jugador(@nivel)
                2. enemigo
        4. nombreJugador

Classes:

1. Main

   Funciones:

   1. Main()

      Primero realiza la conexión con la base de datos y despues muestra un menu con diferentes opciones:

      1. Jugar

         Permite jugar al juego original llamando a la funcion: jugar()

      2. Ver partidas almacenadas

         Muestra las partidas almacenadas con la funcion: Consultas.mostrarPartidas()

      3. Ver combates de una partida

         Permite seleccionar una partida y mostrar los combates guardados en ella con la consulta: Consultas.mostrarCombatePartidas()

      4. Ver jugadores almacenados

         Muestra los jugadores almacenados: Consultas.mostrarJugadores()

      5. Buscar jugador

         Permite buscar un jugador por el nombre con la consulta: Consultas.buscarJugador(nombre)

      6. Cargar jugador

         Permite buscar y cargar un jugador por el nombre con la consulta: Consultas.cargarJugador()
         (Esta funcion en el juego te carga el jugador para usarlo en el juego, usada en este menu fuera de el juego no tiene utilidad)

      7. Crear jugador

         Permite crear un nuevo jugador usando la funcion: crearJugador()

      8. Crear partida

         Selecciona un jugador con la consulta: Consultas.cargarJugador() y despues usa ese jugador para crear y guarda una partida con la funcion: Consultas.guardarPartida()
         (La partida creada no tendra ningun combate almacenado)
         (Las partidas deben ir asociadas de un jugador, por eso se selecciona uno)

      9. Modificar jugador

         Permite buscar un jugador por el nombre y cambiar varios valores predefinidos con la consulta: Consultas.modificarJugador()

      10. Borrar jugador

          Permite buscar y borrar un jugador por el nombre con la consulta: Consultas.borrarJugador()

11. Salir
    Cierra el programa

Este menu esta en un bucle for, hasta que no se seleccione la opcion salir se volvera a mostrar una vez acabada la operación seleccionada.

2. jugar()
   Primero se intenta importar el fichero de los enemigo y si no se puede lo creara.
   Despues se le mostrara al usuario las opciones de nuevo jugador: crear uno nuevo o importarlo, si decide crear uno nuevo se llamara a la funcion crearJugador() , si decide importar llamara a la funcion cargarJugador(), si hay un error al importar mostrara un mensaje de error y las opciones de nuevo jugador de nuevo.
   Mostrara los datos de el jugador para informar al usuario.
   Se creara el objeto partida.
   Para los combates se entrara en un bucle, primero se llamara a la funcion seleccionarEnemigo(enemigos) para seleccionar un enemigo que se le pasara a la funcion combate(jugador, enemigoSeleccionado, partida) esta devolvera true o false para saber si el usuario ha ganado o perdido. Si ha perdido se registrara la derrota en el objeto partida y se terminara el bucle, si ha ganado se registrara el resultado en la partida y si el enemigo es el aleatorio añadira 100 puntos extra por la dificultad de este, ademas le subira el nivel a el jugador, despues comprobara si el nivel es un multiple de 5 entonces le dara una mejora al usuario, se puede elegir entre tener 10 mas de vida o mas 5 de ataque.
   Fuera de el bucle (Solo se puede salir perdiendo) se guardan los datos de la partida y se le muestran al usuario, se le da la opcion de exportar esa partida con la funcion guardarPartida(Partida partida), y despues se le da la opcion de exportar el jugador con la funcion guardarJugador(Jugador jugador).

3. Enemigo seleccionarEnemigo(Enemigo[])
   Recibe la lista de enemigos, los printea y pide al usuario que elja uno, al final devuelve el enemigo seleccionado. Ademas muestra un enemigo aleatorio que no esta en la lista, si el usuario selecciona este enemigo esta misma funcion crea un enemigo nuevo con variables aleatorias y lo devuelve.

4. Boolean combate(Jugador, Enemigo, Partida)
   Recibe el jugador actual y el enemigo seleccionado por la funcion seleccionarEnemigo(), entra en un bucle hasta que se termine el combate, en el combate el primer turno sera de el jugador, tendra la opcion de atacar o curarse, una vez acabado su turno se realizara la operacion, sumar vida al jugador o restarle al enemigo y le sumara puntuacion por la operacion, despues se comprobara si al enemigo le queda vida, si no es asi el combate terminara, se saldra de el bucle y se retornara true indicando que ha ganado, si no sera el turno de el enemigo, este siempre atacara, se le resta la vida al jugador y se comprueba si esta llega a 0, si es asi el combate terminara y se devolvera false indicando que ha perdido y terminando el programa.

5. Jugador crearJugador()
   Pide un nombre al usuario y crea un nuevo objeto jugador con ese nombre y los datos por defecto (Vida = 100, Ataque = 25), y preguntara al usuario si desea exportarlo. Al final devolvera el objeto jugador.

2. Partida
    Variables:
    1. LocalDateTime fecha
    2. int puntuacionTotal
    3. String[] combates
    4. String nombreJugador

3. Jugador
    Variables:
    1. String nombre
    2. int vida
    3. int ataque
    4. int nivel

4. Enemigo
    Variables:
    1. String nombre
    2. int vida
    3. int ataque

    Funciones:
    4. Enemigo[] importarEnemigosDAT()
       Crea una lista de objetos Enemigo vacia, busca el fichero Enemigos.dat . Si el fichero existe creara los flujos y ira leyendo el fichero en la lista, si no existe mostrara un mensaje de error y dejara la lista vacia. Al final devolvera la lista.

    5. void exportarEnemigosDAT(Enemigo[])
       Crea el fichero Enemigos.dat, crea los fujos, va escribiendo los enemigos de la lista que ha recibido y muestra un mensaje cuando termina.

5. Consultas

Variables:

1. String driver
2. String URI
3. String usuario
4. String password
5. Collection col

Funciones:

6. Collection conectar()

   Realiza la conexión a la BBDD eXist, devuelve la cloeccion.

7. void mostrarJugadores()

   Realiza una consulta para mostrar todos los jugadores guardados en el XML de jugadores, formatea el resultado con un concat para que se vea bonito.

8. boolean buscarJugador(String nombre)

   Solicita el nombre de el jugador a buscar despues realiza una consulta para mostrar los jugadores donde pone la condicion que el nombre coincida con el anteriormente indicado, formatea el resultado con un concat para que se vea bonito.

9. Jugador cargarJugador()

   Muestra todos los jugadores llamando a la funcion mostrarJugadores() y solicita el nombre de el jugador a cargar despues realiza una consulta para mostrar los jugadores donde pone la condicion que el nombre coincida con el anteriormente indicado, formatea el resultado con un concat para que los datos lleguen ordenados separados por comas, luego sesepara el string por las comas para tener los diferentes datos en strings que se usan para crear un nuevo objeto Jugador.

10. void mostrarPartidas()

    Realiza una consulta para mostrar todos las partidas guardadas en el XML de partidas, formatea el resultado con un concat para que se vea bonito.

11. void mostrarCombatePartidas()

    Muestra todos las partidas llamando a la funcion mostrarPartidas() y solicita la fecha de la partida a seleccionar, despues realiza una consulta para mostrar los combates donde pone la condicion que la fecha coincida con la anteriormente indicada, formatea el resultado con un concat para que se vea bonito.

12. void guardarJugador(Jugador jugador)

    Recive un jugador, prepara una consulta con los datos de ese jugador y la inserta.

13. void guardarPartida(Partida partida)

    Recive una partida, debido a que la cantidad de combates puede variar en gran medida se preparan en un bucle for donde se va haciendo un string con los combates en el formato de XML, luego prepara una consulta con los demas datos de esa partida y los combates preparados antes y la inserta.

14. void modificarJugador()

    Muestra todos los jugadores llamando a la funcion mostrarJugadores() y solicita el nombre de el jugador a editar despues pregunta por el dato a cambiar (nivel / vida / ataque) y el nuevo valor de ese dato, con estos datos realiza una consulta update donde pone la condicion que el nombre coincida con el anteriormente indicado.

15. void borrarJugador()

    Muestra todos los jugadores llamando a la funcion mostrarJugadores() y solicita el nombre de el jugador a cargar despues realiza una consulta delete donde pone la condicion que el nombre coincida con el anteriormente indicado.