

Proyecto: Elige tu propia aventura refactorizado

Análisis y toma de decisiones:

En un inicio analizamos la versión antigua de nuestro proyecto dándonos cuenta de que necesitábamos repartir los métodos y sus funcionalidades en diferentes clases.

Para ello empezamos haciendo el diagrama de clases para poder hacer una versión funcional del proyecto pero sin base de datos. Después realizamos un boceto de cómo sería nuestra base de datos según las necesidades de cada clase.

Durante el desarrollo de la versión demo tuvimos que añadir y cambiar la lógica de gran parte del proyecto anterior. Los ejemplos más destacables son:

-El Main ya no posee lógica en su interior simplemente crea una instancia de Juego y la utiliza.

```
public static void main(String[] args) throws SQLException {  
  
    Juego juego = new Juego();  
    juego.comenzar();  
  
}
```

-Juego sigue el patrón de diseño singleton, el cual asegura que solo haya una instancia de este. Esto mismo se aplica a las clases que interactúan con la base de datos.

```
public static Juego getInstanceJuego() throws SQLException {  
    if (instanciaJuego == null) { //si no se ha creado el juego, se crea  
        instanciaJuego = new Juego();  
    }  
    return instanciaJuego; //si ya está creado  
}
```

-El Inicio de sesión es claramente diferente, constando ahora de dos posibilidades: guardar un nuevo perfil o iniciar sesión con uno ya existente. La explicación en profundidad se encuentra en el apartado de la incorporación de la base de datos. Aquí mostramos un ejemplo del funcionamiento lógico.

1: Preguntamos el tipo de inicio

```
System.out.println("Bienvenido desconocido usuario, elija su forma de inicio: \n 1.Nuevo Perfil | 2.Perfil existente");  
decision = sc.nextLine();
```

2:En caso de nuevo perfil = Comprobamos que el nombre no esté en la base de datos

```
do {  
  
    System.out.println("\nNombre:");  
    nombreJugador = sc.nextLine();  
    if (perfil.comprobarNombre(nombreJugador) == false) {  
        System.out.println("Ese perfil ya existe\nIniciar sesion: 1.No 2.Si");  
        decision = sc.nextLine();  
    }  
  
}while(perfil.comprobarNombre(nombreJugador) == false && !decision.equals("2"));
```

3: Solicitamos el resto de datos y guardamos el personaje haciendo uso de nuestro método de guardado (explicado en el apartado de base de datos).

```
if (ldecision.equals("2")) {
    System.out.println("Contraseña:");
    contrasena = sc.nextLine();

    do {
        try {
            System.out.println("Elija un valor del 1-100:");
            idAscii = sc.nextInt();
            AsciiBien = true;
        } catch (Exception e) {
            System.out.println("El valor debe ser un numero");
            AsciiBien = false;
        }
    } while (AsciiBien == false);

    perfil.guardarJugador(nombreJugador, contrasena, idAscii, 0, "");
}
```

4: Hacemos un inicio de sesión para probar el funcionamiento del guardado, si se eligió iniciar sesión el programa salta hasta aquí. (El método inicioSesion está explicado en el apartado de base de datos)

```
if (decision.equals("2")) {
    System.out.println("\nBienvenido de nuevo pues, rellene sus datos para saber que es usted:\n");
}

System.out.println("Repita su Nombre:");
nombreJugador = sc.nextLine();

System.out.println("Repita su Contraseña:");
contrasena = sc.nextLine();

if (perfil.inicioSesion(nombreJugador, contrasena) == false) {
    System.out.println(Color.RED_BOLD_BRIGHT + "El nombre o contraseña son incorrectos" + Color.RESET);
}
```

5: Si todo ha salido bien hará una impresión de los datos del jugador por pantalla

```
System.out.println("\nGenial, este eres tu\n" + Color.YELLOW_BOLD_BRIGHT + nombreJugador.toUpperCase() + Color.RESET);
ASCII.printAscii(perfil.getAscii(nombreJugador));
```

```
Bienvenido desconocido usuario, elija su forma de inicio:
1.Nuevo Perfil | 2.Perfil existente
2

Bienvenido de nuevo pues, rellene sus datos para saber que es usted:

Repita su Nombre:
lol
Repita su Contraseña:
lol
Genial, este eres tu
LOL
```



-Hay un nuevo método encargado de mostrar la tabla de puntuaciones de los jugadores, sacada de la tabla jugador de la base de datos. Su funcionalidad se resume en un bucle que extrae e imprime los datos de cada jugador en orden descendente usando formatos de impresión de consola como "%-20s %-15s %-15s\n" para organizar los datos.

```
public void leaderboard(String nombreJugador) throws SQLException {
    String query = "SELECT nombre, puntuacion, rank FROM jugador ORDER BY puntuacion DESC";
    PreparedStatement statement = conn.prepareStatement(query);
    ResultSet rs = statement.executeQuery();
    System.out.println("-----");
    System.out.printf(Color.WHITE_BOLD_BRIGHT + "%-20s %-15s %-15s\n", "Nombre", "Puntuación",
```

```

"Rango" + Color.RESET);
System.out.println("-----");
while (rs.next()) {
    String nombre = rs.getString("nombre");
    int puntuacion = rs.getInt("puntuacion");
    String rank = rs.getString("rank");
    if (nombre.equalsIgnoreCase(nombreJugador)) {
        // Resaltar con asteriscos
        System.out.printf(Color.WHITE_BOLD_BRIGHT + "%-20s %-15d %-15s%n", nombre, puntuacion,
rank + Color.RESET);
    } else {
        System.out.printf("%-20s %-15d %-15s%n", nombre, puntuacion, rank);
    }
}
rs.close();
statement.close();
}

```

-El Bucle de juego se maneja dentro de la propia clase Juego, repitiendo el bucle de mostrar historia, mostrar progresión y mostrar caminos hasta que llegas a la habitación 15 donde te enfrentas al boss.

```

private void bucleJuego() throws SQLException
{
    DecimalFormat dfOneDecimal = new DecimalFormat("0.0");
    DecimalFormat dfZeroDecimal = new DecimalFormat("0");
    while(!gameOver)
    {
        lore();
        System.out.println(prota.getNombre());
        System.out.println("Vida: " + (prota.getVida() > prota.getVidaMax()/2 ?
Color.GREEN_BRIGHT : (prota.getVida() > prota.getVidaMax() / 4 ? Color.YELLOW :
Color.RED_BRIGHT)) + dfOneDecimal.format(prota.getVida()) + Color.RESET + "/" +
Color.GREEN_BRIGHT + dfZeroDecimal.format(prota.getVidaMax()) + Color.RESET
+ " puntos de vida");
        System.out.println("Dinero: " + monedas(prota.getNombre(), prota.getMonedas()) +
Color.YELLOW + (prota.getNombre().equals("Chicken Little") ? " semillas" : "
esmeraldas") + Color.RESET);
        if(numeroSalas > 0) System.out.println(prota.getNombre() + " ha completado " +
numeroSalas + " salas\r\n"); else System.out.println("");
        if(numeroSalas == 14)
        {
            bossBattle();
        }
        else
        {
            seleccionCaminos(random, sc);
        }
        numeroSalas++;
    }
    finales();
}

```

-El método que sirve para seleccionar los caminos el cual anteriormente llamaba a varios métodos y variables dentro de su misma. Ahora contiene un switch que dependiendo de la elección llama a métodos de otras clases como Combate o Evento que desarrollan sus propias funciones.

```

switch(seleccionStr) { // Ir al camino (evento) elegido
    case "Combate":
    {
        if(combate.combate(sc, random, prota, "Normal", random.nextBoolean()))
        {
            calculoFinCombate();
        }
    }
}

```

```

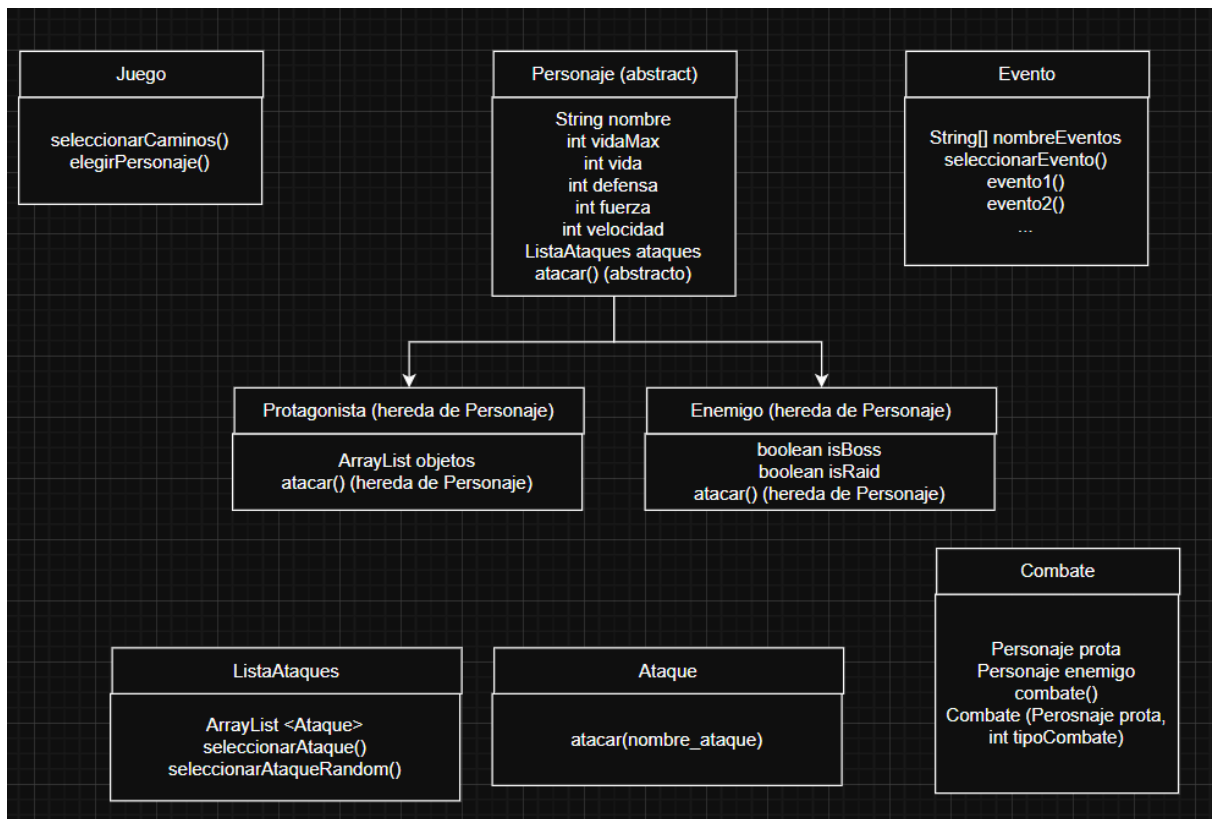
        }
        else
        {
            gameOver = true;
        }
        break;
    }
    case "Evento Aleatorio":
    {
        evento.elegirEvento(prota);
        break;
    }
    case "Mercader":
    {
        if(prota.getBadOmen())
        {
            System.out.println(prota.getNombre() + " se acerca a una aldea
            cercana para comprar cosas con un mercader");
            System.out.println("Pero...");
            System.out.println(prota.getNombre() + " se olvida de que tenía "
            + Color.BLACK_BOLD_BRIGHT + "Mal Presagio" +
            Color.BLACK_BOLD_BRIGHT + "...");
            System.out.println("Una raid va a comenzar pronto!\n\n");
            if(combate.combate(sc, random, prota, "Raid",
            random.nextBoolean()))
            {
                calculoFinCombate();
            }
            else
            {
                gameOver = true;
            }
        }
        else
        {
            evento.eventoMercader(prota, scanner, random);
        }
        break;
    }
}

```

Proceso de refactorización:

1. Segmentamos el código en clases repartiendo responsabilidades.

Diagrama de clases:



2. Expandimos el diagrama de casos de uso.



Uno de los cambios es casos de uso. El anterior no tenía tanta información. Este tiene más y es más acertado a uno real.

3. Incorporación de base de datos

Requisitos:

- **Almacenar y Seleccionar personajes**

Métodos encargados de insertar nuevos personajes

```
public void insertProta(String nombre, Double vidaMax, Double defensa,
Double fuerza, Double velocidad, Integer idAscii, Boolean karma) throws
SQLException {
    String insertSql = "INSERT INTO personajes (nombre, tipo, vidaMax,
defensa, fuerza, velocidad, idAscii, karma, isRaid, isBoss, isSteve) "
        + "VALUES (?, 'protagonista', ?, ?, ?, ?, ?, NULL, NULL,
NULL)";
    PreparedStatement statement = conn.prepareStatement(insertSql);
    statement.setString(1, nombre);

    // Verifica si los valores no son nulos antes de insertarlos
    if (vidaMax != null) statement.setDouble(2, vidaMax); else
statement.setNull(2, java.sql.Types.DOUBLE);
    if (defensa != null) statement.setDouble(3, defensa); else
statement.setNull(3, java.sql.Types.DOUBLE);
    if (fuerza != null) statement.setDouble(4, fuerza); else statement.setNull(4,
java.sql.Types.DOUBLE);
```

```

        if (velocidad != null) statement.setDouble(5, velocidad); else
        statement.setNull(5, java.sql.Types.DOUBLE);
        if (idAscii != null) statement.setInt(6, idAscii); else statement.setNull(6,
        java.sql.Types.INTEGER);
        if (karma != null) statement.setBoolean(7, karma); else
        statement.setNull(7, java.sql.Types.BOOLEAN);
        // Ejecuta la inserción
        int rowsInserted = statement.executeUpdate();
        // Muestra resultado
        if (rowsInserted > 0) {
            System.out.println("Protagonista insertado correctamente");
        } else {
            System.out.println("No se insertó el protagonista");
        }
        statement.close();
    }

```

```

    public void insertEnemigo(String nombre, Double vidaMax, Double defensa,
    Double fuerza, Double velocidad, Integer idAscii, Boolean isRaid, Boolean isBoss,
    Boolean isSteve) throws SQLException {
        String insertSql = "INSERT INTO personajes (nombre, tipo, vidaMax,
        defensa, fuerza, velocidad, idAscii, karma, isRaid, isBoss, isSteve) "
            + "VALUES (?, 'enemigo', ?, ?, ?, ?, NULL, ?, ?,
        ?)";
    }

```

```

        PreparedStatement statement = conn.prepareStatement(insertSql);
        statement.setString(1, nombre);
    }

```

```

        if (vidaMax != null) statement.setDouble(2, vidaMax); else
        statement.setNull(2, java.sql.Types.DOUBLE);
        if (defensa != null) statement.setDouble(3, defensa); else
        statement.setNull(3, java.sql.Types.DOUBLE);
        if (fuerza != null) statement.setDouble(4, fuerza); else
        statement.setNull(4, java.sql.Types.DOUBLE);
        if (velocidad != null) statement.setDouble(5, velocidad); else
        statement.setNull(5, java.sql.Types.DOUBLE);
        if (idAscii != null) statement.setInt(6, idAscii); else
        statement.setNull(6, java.sql.Types.INTEGER);
    }

```

```

    // karma es NULL para enemigo
    if (isRaid != null) statement.setBoolean(7, isRaid); else
    statement.setNull(7, java.sql.Types.BOOLEAN);
    if (isBoss != null) statement.setBoolean(8, isBoss); else
    statement.setNull(8, java.sql.Types.BOOLEAN);
    if (isSteve != null) statement.setBoolean(9, isSteve); else
    statement.setNull(9, java.sql.Types.BOOLEAN);
}

```

```

    int rowsInserted = statement.executeUpdate();
}

```

```

    if (rowsInserted > 0) {
        System.out.println("Enemigo insertado correctamente");
    } else {
    }
}

```

```

        System.out.println("No se insertó el enemigo");
    }

    statement.close();
}

```

Metodos encargados de obtener las estadisticas de los personajes y devolverlas en un objeto personaje llamado dataProta o dataEnemigo

```

// Obtiene los datos del protagonista como objeto
public Protagonista getDataProta(String nombre) throws SQLException {
    String query = "SELECT * FROM personajes WHERE nombre = ? AND tipo = 'protagonista'";
    PreparedStatement statement = conn.prepareStatement(query);
    statement.setString(1, nombre);
    ResultSet rs = statement.executeQuery();
    if (rs.next()) {
        dataProta.setNombre(nombre);
        dataProta.setVidaMax(rs.getDouble("vidaMax"));
        dataProta.setDefensa(rs.getDouble("defensa"));
        dataProta.setFuerza(rs.getDouble("fuerza"));
        dataProta.setVelocidad(rs.getDouble("velocidad"));
        dataProta.setIdAscii(rs.getInt("idAscii"));
        dataProta.getAtaqueController().rellenarAtaques(dataProta);
    } else {
        System.out.println("No se encontró el protagonista: " + nombre);
        return null;
    }
    rs.close();
    statement.close();
    return dataProta;
}

// Obtiene los datos del enemigo como objeto
public Enemigo getDataEnemigo(String nombre) throws SQLException {
    String query = "SELECT * FROM personajes WHERE nombre = ? AND tipo = 'enemigo'";
    PreparedStatement statement = conn.prepareStatement(query);
    statement.setString(1, nombre);
    ResultSet rs = statement.executeQuery();
    if (rs.next()) {
        dataEnemigo.setNombre(nombre);
        dataEnemigo.setVidaMax(rs.getDouble("vidaMax"));
        dataEnemigo.setDefensa(rs.getDouble("defensa"));
        dataEnemigo.setFuerza(rs.getDouble("fuerza"));
        dataEnemigo.setVelocidad(rs.getDouble("velocidad"));
        dataEnemigo.setIdAscii(rs.getInt("idAscii"));
        dataEnemigo.getAtaqueController().rellenarAtaques(dataEnemigo);
    } else {
        System.out.println("No se encontró el enemigo: " + nombre);
        return null;
    }
}

```



```

    }
    rs.close();
    statement.close();
    return dataEnemigo;
}

```

- **Almacenar y Seleccionar ataques**

Mediante el siguiente método creamos nuevos ataques

```

public void insertAtaque(int id, String nombre, String desc) throws SQLException {
    String sql = "INSERT INTO ataques (id, nombre, descripcion) VALUES (?, ?, ?)";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setInt(1, id);
    ps.setString(2, nombre);
    ps.setString(3, desc);
    int result = ps.executeUpdate();
    if (result > 0) {
        System.out.println("Ataque insertado correctamente.");
    } else {
        System.out.println("No se pudo insertar el ataque.");
    }
    ps.close();
}

```

Mediante el siguiente método seleccionamos los ataques según su relación entre el nombrePersonaje y el ataque_id que son columnas de la tabla relacion_ataques.

```

public ArrayList<String> selectAllAtaques(String nombrePersonaje) throws SQLException {
    ArrayList<String> listaNombresAtaques = new ArrayList<>();
    // Consulta para obtener el id del ataque y tipo del personaje
    String query = ""
        SELECT relacion_ataques.ataque_id, personajes.tipo
        FROM relacion_ataques
        JOIN personajes ON relacion_ataques.personaje_nombre =
        personajes.nombre
        WHERE relacion_ataques.personaje_nombre = ?
        """;
    PreparedStatement stmt = conn.prepareStatement(query);
    stmt.setString(1, nombrePersonaje);
    ResultSet rs = stmt.executeQuery();
    ArrayList<Integer> ataqueIds = new ArrayList<>();
    String tipo = null;

    // Guardar los ID de ataques y el tipo del personaje
    while (rs.next()) {
        ataqueIds.add(rs.getInt("ataque_id"));
        tipo = rs.getString("tipo");
    }
    rs.close();
    stmt.close();
}

```

```

// Lógica según el nombre y tipo
// Agrega ataques especiales dependiendo del tipo de personaje
if ("protagonista".equalsIgnoreCase(tipo)) {
    if ("Alex".equalsIgnoreCase(nombrePersonaje)) {
        ataquesIds.add(2); // Huir
    } else {
        ataquesIds.add(1); // Cura
        ataquesIds.add(2); // Huir
    }
}
// Consulta para obtener el nombre del ataque a partir de sus ID
String queryAtaque = "SELECT nombre FROM ataques WHERE id = ?";
PreparedStatement stmtAtaque = conn.prepareStatement(queryAtaque);
for (int ataqueId : ataquesIds) {
    stmtAtaque.setInt(1, ataqueId);
    ResultSet rsAtaque = stmtAtaque.executeQuery();
    if (rsAtaque.next()) {
        listaNombresAtaques.add(rsAtaque.getString("nombre"));
    }
    rsAtaque.close();
}
stmtAtaque.close();
return listaNombresAtaques;
}

```

- **Almacenar Jugadores nuevos e iniciar a los nuevos:**

Usando un método comprobacionNombre, que indica si el nombre ya existe nos aseguramos de poder guardar el jugador.

```

public boolean comprobarNombre(String nombreJugador) throws SQLException {
    String sql = "SELECT * FROM jugador WHERE nombre = ?";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setString(1, nombreJugador);
    ResultSet rs = ps.executeQuery();
    boolean nombreDisponible = !rs.next();
    rs.close();
    ps.close();
    return nombreDisponible;
}

```

Despues mediante la siguiente logica guardamos el jugador en la tabla

```

if (comprobacionNombre) {
    String sql = "INSERT INTO jugador (nombre, contraseña, idAscii, puntuacion, rank) VALUES (?, ?, ?, ?, ?)";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setString(1, nombre);
    ps.setString(2, contraseña);
    ps.setInt(3, idAscii);
    ps.setInt(4, puntuacion);
    ps.setString(5, rank);
    int result = ps.executeUpdate();
    if (result < 0) {
        System.out.println("No se pudo insertar el jugador.");
    }
}

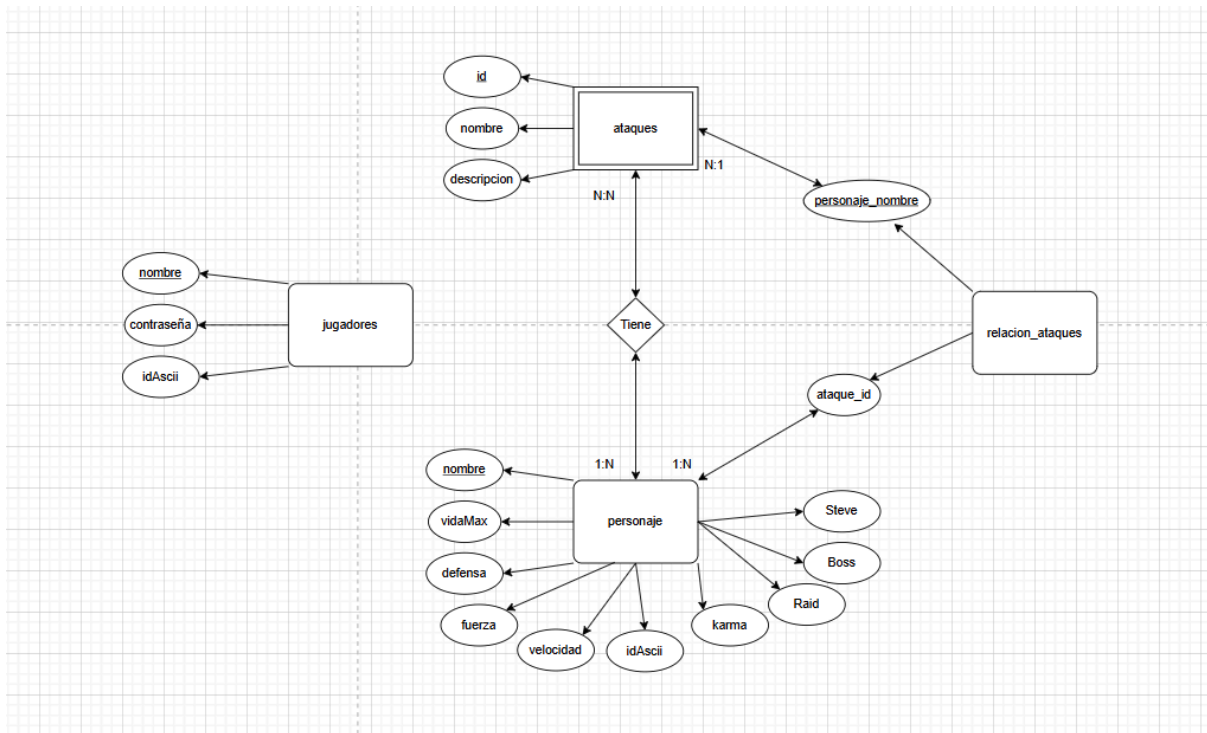
```

```

    }
    ps.close();
} else {
    System.out.println("El nombre está cogido.");
}

```

Diagrama E-R:



Tablas:











































Jugador:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 nombre	varchar(50)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	2 contraseña	varchar(24)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 idAscii	int(11)			Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/>	4 Puntuacion	int(11)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	5 Rank	varchar(50)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más







	nombre	contraseña	idAscii	Puntuacion	Rank
<input type="checkbox"/> Editar Copiar Borrar			99	-188	ZZZ-RANK
<input type="checkbox"/> Editar Copiar Borrar	Aitana	1234	NULL	158	C-RANK
<input type="checkbox"/> Editar Copiar Borrar	Bombardeen a Diego	BombardeenADiego	NULL	1393	SS-RANK
<input type="checkbox"/> Editar Copiar Borrar	fran	flan	99	978	A-RANK
<input type="checkbox"/> Editar Copiar Borrar	John		5	1017	S-RANK
<input type="checkbox"/> Editar Copiar Borrar	lol	lol	3	-300	ZZZ-RANK
<input type="checkbox"/> Editar Copiar Borrar	Luna	1234	NULL	1616	SSS-RANK
<input type="checkbox"/> Editar Copiar Borrar	Null	null	3	0	ZZZ-RANK
<input type="checkbox"/> Editar Copiar Borrar	Pelayo	1234	11	1113	S-RANK

Personajes:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 nombre	 varchar(50)	utf8mb4_general_ci		No	Ninguna			 Cambiar  Eliminar Más
<input type="checkbox"/>	2 tipo	enum('protagonista', 'enemigo')	utf8mb4_general_ci		No	Ninguna			 Cambiar  Eliminar Más
<input type="checkbox"/>	3 vidaMax	double			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	4 defensa	double			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	5 fuerza	double			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	6 velocidad	double			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	7 idAscii	int(11)			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	8 karma	tinyint(1)			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	9 isRaid	tinyint(1)			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	10 isBoss	tinyint(1)			Sí	NULL			 Cambiar  Eliminar Más
<input type="checkbox"/>	11 isSteve	tinyint(1)			Sí	NULL			 Cambiar  Eliminar Más

			nombre	1	tipo	vidaMax	defensa	fuerza	velocidad	idAscii	karma	isRaid	isBoss	isSteve
<input type="checkbox"/>	 Editar  Copiar  Borrar	Alex	protagonista	30	0	0	1	3	1	NULL	NULL	NULL		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Chicken Little	protagonista	15	0	0	3.5	4	0	NULL	NULL	NULL		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Coronel Sanders	enemigo	50	5	0	5	0	NULL	0	1	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Creeper	enemigo	15	0	0	5	0	NULL	0	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Dragon	enemigo	50	5	0	5	0	NULL	0	1	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Esqueleto	enemigo	15	0	0	2	0	NULL	0	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	EvilSteve	enemigo	30	5	0	5	0	NULL	0	0	1		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Evoker	enemigo	15	0	0	4	0	NULL	1	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Pillager	enemigo	20	1	0	3	0	NULL	1	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Ravager	enemigo	30	4	0	2	0	NULL	1	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Sans	enemigo	1	0	0	10	0	NULL	0	1	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Steve	protagonista	20	0	1	3	2	0	NULL	NULL	NULL		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Vindicator	enemigo	20	2	0	3	0	NULL	1	0	0		
<input type="checkbox"/>	 Editar  Copiar  Borrar	Zombie	enemigo	20	1	0	1	0	NULL	0	0	0		

Ataques:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 id	int(11)			No	Ninguna		AUTO_INCREMENT	 Cambiar  Eliminar Más
<input type="checkbox"/>	2 nombre	varchar(50)	utf8mb4_general_ci		No	Ninguna			 Cambiar  Eliminar Más
<input type="checkbox"/>	3 descripcion	varchar(500)	utf8mb4_general_ci		No	Ninguna			 Cambiar  Eliminar Más

		id	nombre	descripcion
<input type="checkbox"/>	 Editar  Copiar  Borrar	1	Curar	Ataque de sanación exclusivo para protagonistas
<input type="checkbox"/>	 Editar  Copiar  Borrar	2	Huir	Ataque para escapar exclusivo para protagonistas
<input type="checkbox"/>	 Editar  Copiar  Borrar	3	Por si Espada	Ataque básico con espada
<input type="checkbox"/>	 Editar  Copiar  Borrar	4	Arco	7 de daño, pero tienes que recargar para usar otra...
<input type="checkbox"/>	 Editar  Copiar  Borrar	5	Ataque Crítico	0-15 de daño
<input type="checkbox"/>	 Editar  Copiar  Borrar	8	Hacha	5 de daño
<input type="checkbox"/>	 Editar  Copiar  Borrar	9	Poción de Fuerza	Incrementa el daño del siguiente ataque en +4 de d...
<input type="checkbox"/>	 Editar  Copiar  Borrar	10	Maldición de Wither	Reduce la vida del enemigo en un 20% de su vida ac...
<input type="checkbox"/>	 Editar  Copiar  Borrar	12	Tricotada	2 de daño x 3 veces
<input type="checkbox"/>	 Editar  Copiar  Borrar	13	Ataque huevo	4 de daño y +1 pollo
<input type="checkbox"/>	 Editar  Copiar  Borrar	14	Llamada de pollos	Te cubres de pollos y la siguiente vez que te ataq...
<input type="checkbox"/>	 Editar  Copiar  Borrar	32	Puñetazo	Golpe básico con el puño
<input type="checkbox"/>	 Editar  Copiar  Borrar	33	Mordisco	Mordisco que puede causar infección
<input type="checkbox"/>	 Editar  Copiar  Borrar	34	Infección	Causa daño progresivo
<input type="checkbox"/>	 Editar  Copiar  Borrar	35	Puñetazo	Golpe básico con el puño
<input type="checkbox"/>	 Editar  Copiar  Borrar	36	Flecha	Dispara una flecha al objetivo
<input type="checkbox"/>	 Editar  Copiar  Borrar	37	Skele-Ton	Invoca un minion esqueleto
<input type="checkbox"/>	 Editar  Copiar  Borrar	38	Tss	Siseo del creeper antes de explotar
<input type="checkbox"/>	 Editar  Copiar  Borrar	39	Tss Tss	Siseo intensificado del creeper
<input type="checkbox"/>	 Editar  Copiar  Borrar	40	Explosión	Explosión que causa daño en área

Relacion_Atiques:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	personaje_nombre	varchar(50)	utf8mb4_general_ci	No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	2	ataque_id	int(11)		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3	tipo_personaje	enum('protagonista', 'enemigo')	utf8mb4_general_ci	No	Ninguna			Cambiar Eliminar Más

	personaje_nombre	ataque_id	tipo_personaje
<input type="checkbox"/>	Editar Copiar Borrar Alex		8 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Alex		9 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Alex		10 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Chicken Little		12 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Chicken Little		13 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Chicken Little		14 protagonista
<input type="checkbox"/>	Editar Copiar Borrar Coronel Sanders		59 enemigo
<input type="checkbox"/>	Editar Copiar Borrar Coronel Sanders		60 enemigo
<input type="checkbox"/>	Editar Copiar Borrar Coronel Sanders		61 enemigo
<input type="checkbox"/>	Editar Copiar Borrar Creeper		38 enemigo
<input type="checkbox"/>	Editar Copiar Borrar Creeper		39 enemigo
<input type="checkbox"/>	Editar Copiar Borrar Creeper		40 enemigo

Consultas necesarias:

Necesitaremos consultas que nos permitan obtener la información de personajes específicos, o generales. Lo mismo para ataques, además de poder insertar, actualizar y borrar.

Sintaxis de las consultas:

```
String selectAllData = "SELECT * FROM protagonistas";
```

```
String updateSql = "UPDATE personajes SET vidaMax=?, defensa=?, fuerza=?, velocidad=?, idAscii=?, karma=?, isRaid=NULL, isBoss=NULL, isSteve=NULL WHERE nombre=? AND tipo='protagonista'";
```

```
String sql = "DELETE FROM personajes WHERE nombre=?";
```

Nuevas funcionalidades (puzzles, enemigos y batallas) cómo se han integrado en la historia.

No hubo grandes cambios en puzzles, enemigos o batallas.

Aunque un ejemplo sería usar los objetos, que haya una descripción sobre el clima en la batalla contra el jefe final o cálculos más específicos en el daño y cómo afecta a la vida.

Por ejemplo: Un ataque tiene su daño y a eso se lo suma a la fuerza del personaje. El personaje tiene una defensa por lo que se calcula la defensa del personaje con el daño que haya recibido.

También creamos nuevas clases, como juego, eventos u objetos.

Referencias/Recursos Utilizados:

Informacion:

<https://www.w3schools.com/>

Recursos:

Java - Spring Tool Suite 4

<https://spring.io/tools>

Github - GitKraken

<https://github.com/Pelenior/F.R.A.N.-FreeRandomAssNewgame->

SQL - XAMPP

<https://www.apachefriends.org/es/index.html>