



Práctica 3 - MongoDB

Gestión de Datos en Medios Digitales



Grupo 1

Blanca García Vera

Luis Antonio González Martínez

Mario López García

Juan Alessandro Vázquez Bustos

Alejandro Vargas Lugo

Índice

Introducción	4
Consultas más frecuentes	5
Las descripciones de los ítems del jugador con id X.	5
Número de puntos corazón de cada personaje.	5
Número de monedas del personaje.	5
Número de puntos estrella del jugador con id X.	5
Índice de defensa y ataque de cada enemigo.	6
Salud del enemigo con nombre X.	6
El número de ayudantes que tiene una batalla.	6
Número de puntos flor de los que dispone el jugador.	6
Ordenar las habilidades estrella por coste de menor a mayor.	7
Ver si el número de PE que tiene el jugador es inferior a 8.	7
Comprobar si el número de puntos nivel es igual o superior a 100.	7
Las habilidades que tiene el personaje acompañante del jugador.	8
Las habilidades que tiene el personaje acompañante del jugador.	8
Consultar el lugar de procedencia de un personaje acompañante.	8
Consultar el estado del jugador.	8
¿Por qué estas consultas?	9
Análisis	9
Inserción de datos	9
Describiendo la colección Partidas	10
Consultas SQLite traducidas a MongoDB	17
Consultas CRUD SQLite a mDB (simples)	17
Todos los jugadores con más de 50 monedas y 30 puntos flor.	17
Todas las habilidades estrella que consuman más de 4 puntos estrella.	17

Todos los lugares donde haya más de 10 enemigos y 5 NPCs.	17
Consultas CRUD SQLite a mDB (compuestas)	18
Todos los enemigos que vuelan que vivan en X región.	18
La descripción de un ítem cuyo nombre sea X del jugador con id X.	18
Nombre de los enemigos que se han enfrentado al jugador con id X.	18
Consultas CRUD SQLite a mDB (agrupadas)	19
Agrupar las habilidades estrella según su coste y mostrar cuántas habilidades hay de cada coste.	19
Calcular la media de PC de los personajes según su procedencia.	19
Calcular el número total de ítems de los jugadores.	19
Índices	20
Jugadores que pueden subir de nivel	20
Jugadores que pueden abrir la Puerta Milenaria (todas las estrellas)	20
Jugadores que tienen un estado concreto	20
Jugadores que tienen un gran número de puntos flor y de monedas	20
Ordenar las Hab. Estrella según su coste	20
Ordenar por número de monedas	21
Reducir la búsqueda a los enemigos que son débiles	21
Reducir la búsqueda a los enemigos que son boss	21
Evitar nombres duplicados	21
Mostrar jugadores con menos de 4 Puntos Estrella	21
Índice compuesto para consultar por procedencia del acompañante	21
Índice Compuesto para consultar los valores máximos de PC, PF, PE y PM	21
22	
Estudio de rendimiento	22
Recursos y bibliografía	23

Introducción

Esta tercera y última práctica de la asignatura tiene como objetivo la creación de una base de datos completa, junto con varias consultas correspondientes, usando el sistema de bases de datos noSQL MongoDB (<https://www.mongodb.com/>) y tomando como base aquellas realizadas en las dos prácticas anteriores, que se habían realizado con tecnologías basadas en SQL y XML respectivamente, y debían hacerse sobre un juego ya creado anteriormente. El juego seleccionado fue **Paper Mario: La Puerta Milenaria** (Nintendo, 2004), del cual también se ha presentado recientemente un remake (2024). En esta entrega, Mario comienza una misión para conseguir las siete Estrellas de Cristal y abrir la Puerta Milenaria. Además, en su aventura se suma otro obstáculo: la princesa Peach ha sido secuestrada por los X-Nauts y debe ser rescatada. Mario y sus enemigos tienen una gran variedad de habilidades y parámetros, que son los que inspiran nuestra base de datos



Cabecera del recién lanzado remake de *Paper Mario: La Puerta Milenaria*.

Consultas más frecuentes

Estas consultas han sido creadas desde el punto de vista del usuario. Son todas nuevas y creadas para poder adaptar la base de datos del juego a MongoDB:

Las descripciones de los ítems del jugador con id X.

Esta consulta se utiliza cuando se abre el inventario, ya que deben aparecer todos los ítems en el menú.

```
// 1)Las descripciones de los ítems del jugador con id X (cuando se abre el inventario  
// deben aparecer todos los ítems).  
db.Partidas.find({"_id": 'game01'}, {"Items.Descripcion": 1, "_id": 0})
```

Número de puntos corazón de cada personaje.

Aparecen en la interfaz de usuario del juego, por lo que debe accederse al dato constantemente.

```
// 2)Número de puntos corazón de cada personaje (aparecen en la UI del juego,  
// por lo que debe accederse al dato constantemente).  
db.Partidas.find({ "_id": 'game01' }, {CurrentPC: 1, _id: 0});
```

Número de monedas del personaje.

Igual que en la anterior, aparece de forma constante en la interfaz de usuario.

```
// 3)Número de monedas del personaje (aparece en la UI).  
db.Partidas.find({ "_id": 'game01' }, {Monedas: 1, _id: 0});
```

Número de puntos estrella del jugador con id X.

Durante la batalla, los puntos estrella aparecen en la interfaz superior como varios círculos pequeños que se van llenando gracias a las reacciones del público, y se van gastando al usar habilidades estrella.

```
// 4)Número de puntos estrella que posee actualmente el jugador con id X  
// (aparece en la barra azul de la UI).  
db.Partidas.find({"_id": "game03"}, {"_id": 0, PE_Actuales: "$CurrentPE"})
```

Índice de defensa y ataque de cada enemigo.

Esta información se obtiene justo antes de comenzar una batalla. Esta información es invisible al jugador hasta que se usa la habilidad Descripción del acompañante Goomarina.

```
// 5)Índice de defensa y ataque de cada enemigo, ya que se accederá cada vez que haya una batalla.  
db.Partidas.aggregate([  
    {$unwind: "$LogEnemigos"},  
    {$unwind: "$LogEnemigos.Enemigos"},  
    {$project: {"_id": 0, Ataque: "$LogEnemigos.Enemigos.PuntosAtaque",  
              Defensa: "$LogEnemigos.Enemigos.PuntosDefensa"}}  
])
```

Salud del enemigo con nombre X.

En la batalla se irá comprobando continuamente.

```
// 6)Salud del enemigo con nombre X (en la batalla se irá comprobando continuamente).  
db.Partidas.aggregate([  
    {$unwind: "$LogEnemigos"},  
    {$unwind: "$LogEnemigos.Enemigos"},  
    {$match: {"LogEnemigos.Enemigos.Nombre": "Koopa Troopa"}},  
    {$project: {"_id": 0, Salud: "$LogEnemigos.Enemigos.PuntosCorazon"}}  
])
```

El número de ayudantes que tiene una batalla.

Cada vez que haya una batalla se accede al dato para generar los enemigos secundarios.

```
// 7)El número de ayudantes que tiene una batalla (cada vez que haya una batalla se  
// accederá al dato para generar los enemigos secundarios.  
db.Partidas.find({"_id": "game01"}, {"_id": 0, "LogEnemigos.NumAyudas": 1})
```

Número de puntos flor de los que dispone el jugador.

Es necesario mostrarlo durante las batallas para que el jugador sepa qué habilidades puede realizar.

```
// 8)Número de puntos flor de los que dispone el jugador.  
db.Partidas.find({"_id": "game01"}, {"_id": 0, "CurrentPF": 1})
```

Ordenar las habilidades estrella por coste de menor a mayor.

Al usar una habilidad estrella, se deben mostrar en una lista según su coste de menor a mayor.

```
// 9)Ordenar las habilidades estrella por coste de menor a mayor.  
db.Partidas.aggregate([  
  {  
    $unwind: "$HabEstrella" // Descomponer en documentos individuales  
  },  
  {  
    $project: {  
      _id: 0,  
      Nombre: "$HabEstrella.Nombre",  
      CostePE: "$HabEstrella.CostePE",  
      Descripcion: "$HabEstrella.Descripcion"  
    }  
  },  
  {  
    $sort: {  
      CostePE: 1  
    }  
  }  
)
```

Ver si el número de PE que tiene el jugador es inferior a 8.

Ya que el número máximo de PE que puede tener el jugador son 8 ($7 + 1$, la que dan por defecto), cuando llega a 8 y el jugador va a la puerta milenaria, esta se abre.

```
// 10)Comprobar si el número de puntos estrella que tiene el jugador es inferior a 8  
// ( $7 +$  la que dan por defecto).  
// (cuando llega a 8 y el jugador va a la puerta milenaria se abre)  
db.Partidas.find({CurrentPE:{$lt:8}}, {_id:0,Nombre:1,CurrentPE:1})
```

Comprobar si el número de puntos nivel es igual o superior a 100.

Cuando el jugador llega a 100 puntos de nivel, tras el final de una batalla, este sube de nivel y los puntos se vuelven a dejar a 0.

```
// 11)Comprobar si el número de puntos nivel es 100 para subir de nivel al personaje y  
// volver a poner los puntos nivel a 0.  
db.Partidas.find({PuntosNivel:{$gt:100}}, {_id:0,Nombre:1,PuntosNivel:1})
```

Las habilidades que tiene el personaje acompañante del jugador.

Durante el turno del acompañante en las batallas, se deben mostrar las habilidades que este puede realizar.

```
// 12)Las habilidades que tiene el personaje acompañante del jugador.  
db.Partidas.find({}, {_id:0,"Acompañante.Nombre":1,"Acompañante.Habilidades":1})
```

Las habilidades que tiene el personaje acompañante del jugador.

Ya que el jugador no podrá huir de una batalla en la que el enemigo sea un boss, es necesario hacer una comprobación justo antes del comienzo de la batalla.

```
// 13)Saber si el enemigo es un jefe (boss) o no.  
db.Partidas.find({}, {_id:0,Nombre:1,"LogEnemigos.Enemigos.Nombre":1})
```

Consultar el lugar de procedencia de un personaje acompañante.

Es importante saber dónde se puede conseguir el personaje.

```
// 14)Consultar el lugar de procedencia de un personaje acompañante, para saber donde se puede conseguir.  
db.Partidas.find({}, {_id:0,"Acompañante.Nombre":1,"Acompañante.Procedencia.Nombre":1})
```

Consultar el estado del jugador.

Antes de comenzar (reanudar) una partida ya existente, es necesario comprobar el estado en el que se encontraba el jugador antes de cerrar la partida.

```
// 15)Consultar la ultima zona donde ha estado el jugador.  
db.Partidas.find({}, {_id:0,Nombre:1,UltimoSubLugar:1})
```

¿Por qué estas consultas?

Hemos elegido estas consultas como las más frecuentes ya que permiten conocer datos cruciales de cada uno de los elementos del juego. Se han incluido consultas en las que se quiere conocer estadísticas del jugador, como ver los puntos estrella de los dispone el jugador, para saber qué habilidades estrella puede usar y cuáles no, o consultar las monedas que tiene y saber qué puede comprar en la tienda. También hay consultas que permiten saber la situación de la batalla, como conocer el número de ayudantes que tiene el enemigo durante la misma; o consultas que nos permiten conocer datos del enemigo al que nos enfrentamos, como saber si es un boss o no, para determinar si el jugador puede huir de la batalla o no, así como índice de ataque y defensa.

Estas consultas son necesarias ya que son las que sustentan el sistema de combate del juego, además de los datos del jugador e incluso información de la propia partida.

Análisis

MongoDB presenta numerosas diferencias estructurales frente a SQL, siendo una de ellas el uso de **colecciones** en vez de **tablas**, las cuales son más flexibles y permiten el uso de más tipos de datos. Esta flexibilidad nos ha permitido consolidar las once tablas que teníamos en la base de datos original en una única gran colección llamada Partidas. Cada elemento de esta colección contiene toda la información referente a una determinada partida guardada, incluyendo los datos del jugador (nombre), el estado global del juego (tiempo de juego, fecha de último guardado, estado de la partida), el inventario del jugador (puntos corazón, flor y estrella; monedas, piezas estrella, soles...), y un histórico de eventos que se han llevado a cabo durante la partida (batallas, estrellas recogidas, personajes desbloqueados...)

Inserción de datos

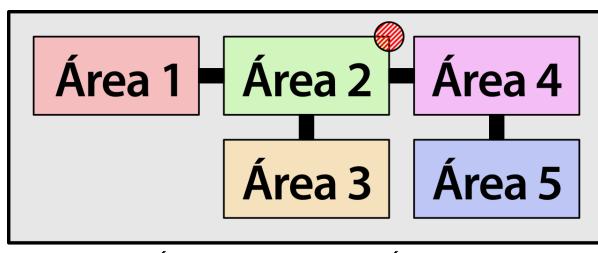
Para la inserción de la información de la base de datos (todos los jugadores con su información correspondiente, acompañantes, enemigos, etc.) se ha seguido un procedimiento similar al de la primera práctica, en la que se ha generado una única gran tabla en Microsoft Excel con muchos elementos con datos e información generada de forma aleatoria (dentro de los rangos permitidos) usando fórmulas muy elaboradas que hacen uso de referencias indirectas. Tras generarla, se han concatenado todas las celdas en una única y se han transferido y formateado en un fichero JSON para su posterior inserción en MongoDB.

_Id	Nombre	Region	PuntosAtaque	PuntosDefensa	PuntosCorazon	Vuela	TienePincho	TieneCaparazon	EsBoss
enemy001	Goomba	{ "\$ref": "Lugares", "\$id": "place22" }	1	0	2	false	false	false	false
enemy002	Paragoomba	{ "\$ref": "Lugares", "\$id": "place22" }	1	0	2	true	false	false	false
enemy003	Goompincho	{ "\$ref": "Lugares", "\$id": "place22" }	2	0	2	false	true	false	false
enemy004	Hiper Goomba	{ "\$ref": "Lugares", "\$id": "place12" }	2	0	8	false	false	false	false
enemy005	Hiper Paragoomba	{ "\$ref": "Lugares", "\$id": "place12" }	2	0	8	true	false	false	false
enemy006	Hiper Goompincho	{ "\$ref": "Lugares", "\$id": "place12" }	3	0	8	false	true	false	false
enemy007	Gloomba	{ "\$ref": "Lugares", "\$id": "place21" }	3	0	7	false	false	false	false
enemy008	Paragloomba	{ "\$ref": "Lugares", "\$id": "place21" }	3	0	7	true	false	false	false

Describiendo la colección Partidas

El _id de cada elemento de la colección tiene el formato “gameXXX” y los atributos son los siguientes:

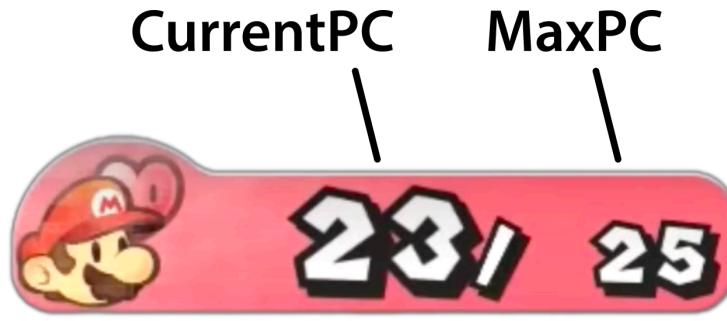
- **Nombre** — nombre o apodo del jugador. Este se elige al comenzar una nueva partida y no se puede cambiar.
- **TiempoJuego** — las horas, minutos y segundos que el jugador ha estado dentro de la partida.
- **Progreso** — porcentaje de completación del juego.
- **UltimaVez** — fecha y hora de la última vez que se guardó la partida.
- **UltimoLugar** — Lugar donde estaba situado el jugador cuando se guardó por última vez. Cada lugar tiene como atributos:
 - **Nombre** — nombre del lugar.
 - **NPCs** — indica el número de NPCs que se encuentran en dicho lugar.
 - **Enemigos** — indica el número de enemigos que se encuentran en el lugar.
- **UltimaArea** — El área específica del lugar donde estaba situado el jugador, que indica el lugar exacto donde debe respawnnear el jugador al meterse de nuevo a la partida.
 - En el siguiente ejemplo, el UltimoLugar es “Gran Árbol”, y dentro de este, el área donde se realizó el último guardado es en el Área 2, sabiendo así el identificador del área exacta donde debe aparecer el jugador de nuevo al volver a acceder a la partida. En el segundo, se puede apreciar una captura del propio juego donde se ve como dicho lugar (Gran Árbol) está dividido en áreas (Planta baja), que para la simplificación de la base de datos, se guarda internamente como un entero (2).



Lugar "Gran Árbol" Guardado en Área 2



- **Estado** — estado en el que se encontraba el jugador la última vez que se guardó la partida.
- **Acompañante** — en las prácticas anteriores recibía el nombre de personaje, se ha cambiado para que no preste a confusión. Indica el acompañante activo en ese momento. Cada acompañante tiene como atributos:
 - **Nombre** — nombre del acompañante.
 - **MaxPC** — número máximo de PC (Puntos Corazón) que puede tener el acompañante. Este, al igual que el jugador, tiene un límite en cuanto a la cantidad de Puntos Corazón que puede tener, y este límite puede aumentar durante el curso de la partida.
 - **CurrentPC** — la cantidad de PC que tiene actualmente (en el momento del último guardado o durante el transcurso de la partida).
 - **Procedencia** — el lugar donde el jugador encuentra (desbloquea) al acompañante. Contiene los tres atributos de Lugar descritos anteriormente.
 - **Descripción** — breve descripción del personaje.
 - **Habilidades** — las habilidades que puede ejecutar el personaje durante las batallas. Cada habilidad a su vez consta de los atributos:
 - **Nombre** — nombre de la habilidad
 - **CostePF** — coste en PF (Puntos Flor) para ejecutar la habilidad.
 - **IndiceAtaque** — los puntos de ataque que infinge la habilidad.
 - **IndiceDefensa** — los puntos de defensa que infinge la habilidad.
 - **Descripción** — breve descripción del acompañante.
- **MaxPC, MaxPF, MaxPE y MaxPM** — indica el número máximo de Puntos Corazón, Flor, Estrella y Medalla, respectivamente. Durante la partida, las estadísticas máximas que puede tener el jugador pueden aumentar. Representa un límite para los diferentes puntos que puede tener el jugador en un preciso momento.
- **CurrentPC, CurrentPF, CurrentPE, y CurrentPM** — indica el número de Puntos Corazón, Flor, Estrella y Medalla, respectivamente, que el jugador tiene actualmente (en el momento del último guardado o durante el transcurso de la partida).



- **Monedas** — necesarias para comprar ítems y medallas, entre otros.
- **PuntosNivel** — cuando el jugador gana una batalla, consigue una cantidad de puntos de nivel que varía en función del número de enemigos y su dificultad relativa al jugador. Cuando este consigue 100 puntos, sube de nivel y puede aumentar sus PC, PF o PM (Puntos Medalla).



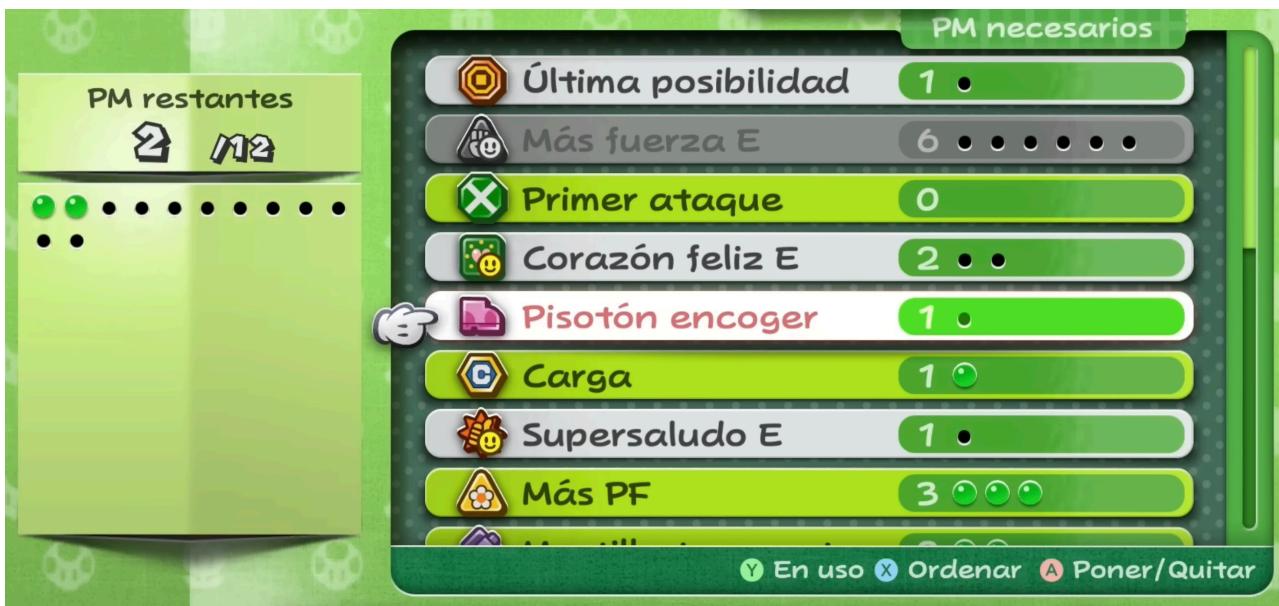
Cuando el jugador alcanza los 100 puntos de nivel, esta variable vuelve a 0 y se puede mejorar una de las estadísticas de la imagen.



Interfaz de Usuario del juego, que muestra las variables del jugador (PC, PF, PE, PuntosNivel y Monedas), así como los PC del personaje acompañante.

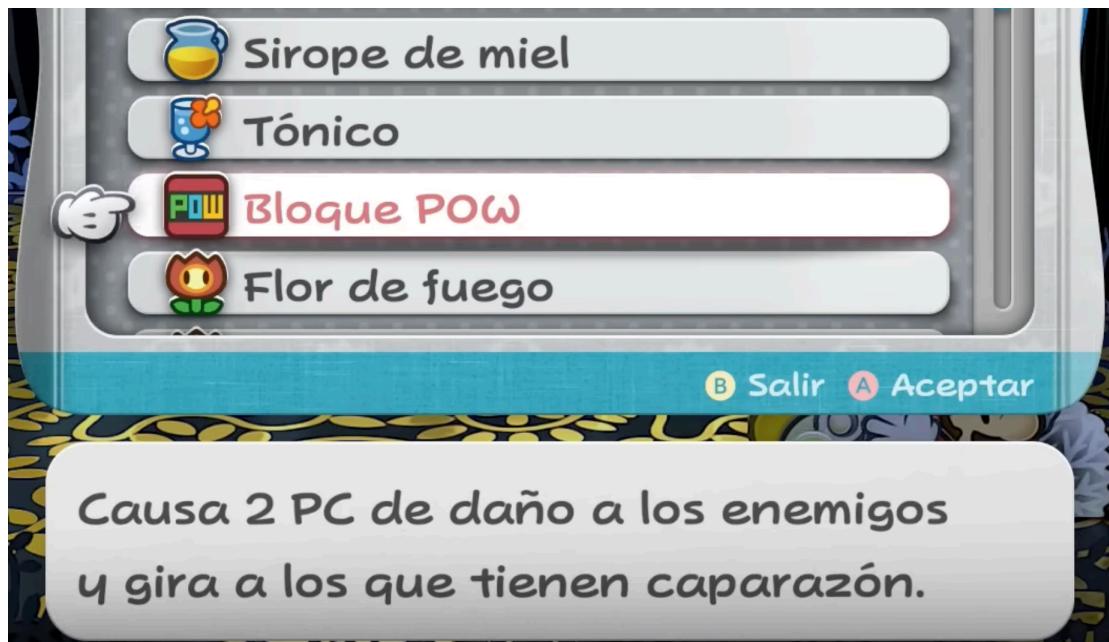
- **PiezasEstrella** — indica el número de piezas estrella que tiene el jugador. Las piezas estrella están repartidas por todo el mundo y se pueden intercambiar por medallas en Villa Viciosa.
- **Soles** — indica el número de soles que tiene el jugador. Los soles están repartidos por todo el mundo y se pueden usar para subir de nivel a los acompañantes.
- **Medallas** — array que indica las medallas que porta el jugador. Cada medalla contiene los siguientes atributos:
 - **Nombre** — nombre de la medalla.
 - **CostePM** — coste en Puntos Medalla (PM) de ponerse (activar) la medalla.
 - **Descripción** — breve descripción de la medalla y sus efectos.

- **EstaActivo** — variable booleana que indica si la medalla está activa o no.



Funcionamiento de las medallas del juego. A la izquierda se muestran los PM de los que dispone el jugador actualmente (2) y los PM totales que tiene (12). Las medallas con fondo verde son las que están activas (se define por la variable EstaActivo), las que tienen fondo blanco son las que se pueden activar (porque se tienen suficientes PM para hacerlo), y las que se muestran en gris son las medallas que no se pueden activar por falta de PM.

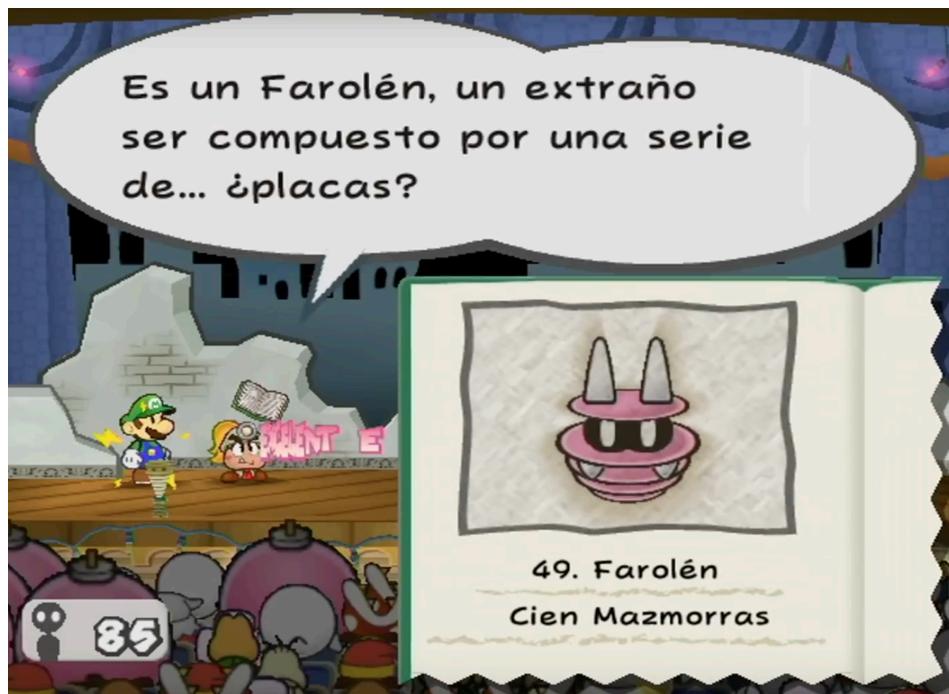
- **Ítems** — array que indica los ítems que porta el jugador. Cada ítem contiene los siguientes atributos:
 - **Nombre** — nombre del ítem
 - **Descripción** — breve descripción del ítem y sus efectos.



Ejemplo de inventario donde se ven los ítems que tiene el jugador en su posesión. Al acceder a un objeto concreto, se muestra la descripción de este.

- **HabEstrella** — array que muestra las Habilidades Estrella que puede ejecutar el jugador. Estas se van adquiriendo a medida que avanza la historia. Cada Hab. Estrella contiene los siguientes atributos:
 - **Nombre** — nombre de la habilidad.
 - **CostePE** — coste en Puntos Estrella (PE) de ejecutar la habilidad.
 - **Descripción** — breve descripción de la habilidad y de sus efectos.
- **LogEnemigos** — este array sustituye a la tabla Batallas. Contiene un histórico de todas las batallas a las que se ha enfrentado el jugador. Cada batalla contiene los siguientes atributos:
 - **NumBatalla** — índice de la batalla. La primera batalla tendrá índice 1, la segunda 2, y así sucesivamente.
 - **NumEnemigos** — número de enemigos que participan en la batalla.
 - **NumAyudas** — número de ayudas durante la batalla.
 - **Enemigos** — array que contiene tantos elementos como se hayan definido en NumEnemigos. Cada enemigo contiene los siguientes atributos:
 - **EnemigoID** — ID del enemigo. Dentro del juego cada enemigo tiene su propio ID numérico. Mediante el apartado Descripción del menú del juego, el jugador puede acceder a un bestiario donde aparecen todos los enemigos a los que se ha enfrentado ordenados por ID.
 - **Nombre** — nombre del enemigo.
 - **Region** — lugar donde se puede encontrar el enemigo.

- **PuntosAtaque** — número de puntos de ataque que tiene el enemigo.
- **PuntosDefensa** — número de puntos de defensa que tiene el enemigo.
- **PuntosCorazon** — número de PC (Puntos Corazón) que tiene el enemigo.
- **Vuela, TienePincho, TieneCaparazon y EsBoss** — cuatro variables booleanas que definen si el enemigo puede volar (por lo que no le afectarían los ataques terrestres), tiene pinchos (por lo que no se le podría saltar encima), tiene caparazón (por lo que no le afectarían los ataques débiles) y si es un boss (por lo que no se podría huir de la batalla con el mismo).



Entrada en el bestiario del juego de un enemigo concreto. Esta entrada contiene su ID que le identifica (49), el nombre del enemigo (Farolén) y la región en la que se puede encontrar al enemigo (Cien Mazmorras), entre otras cosas.



En esta imagen se puede ver que, como el jugador se está enfrentando a un boss (EsBoss = true), no puede huir de la batalla.

Consultas SQLite traducidas a MongoDB

Son las consultas de las prácticas anteriores adaptadas al contexto de una base de datos no relacional, más específicamente al código de MongoDB. Cada tipo de consulta de SQLite está correctamente identificada con cada una de sus funciones.

Consultas CRUD SQLite a mDB (simples)

Todos los jugadores con más de 50 monedas y 30 puntos flor.

```
// 1. Todos los jugadores con más de 50 monedas y 30 puntos flor.  
db.Partidas.find({Monedas:{$gt:50}, CurrentPF:{$gt:30}})
```

Todas las habilidades estrella que consuman más de 4 puntos estrella.

```
// 2. Todas las habilidades estrella que consuman más de 4 puntos estrella.  
db.Partidas.aggregate([  
  {  
    $unwind: "$HabEstrella" // Descomponer el array "HabEstrella" en documentos individuales  
  },  
  {  
    $match: {  
      "HabEstrella.CostePE": { $gt: 4 } // Filtrar habilidades estrella con un coste mayor a 1  
    }  
  },  
  {  
    $project: {  
      _id: 0,  
      Nombre: "$HabEstrella.Nombre",  
      CostePE: "$HabEstrella.CostePE",  
      Descripcion: "$HabEstrella.Descripcion"  
    }  
  }  
])
```

Todos los lugares donde haya más de 10 enemigos y 5 NPCs.

```
// 3. Todos los lugares donde haya más de 10 enemigos y 5 NPCs.  
db.Partidas.aggregate([  
  {$unwind: "$UltimoLugar"},  
  {$match: {$and: [{"UltimoLugar.Enemigos":{$gt:10}}, {"UltimoLugar.NPCs":{$gt:5}}]}},  
  {$project: {"_id": 0, Lugar: "$UltimoLugar.Nombre"}}  
])
```

Consultas CRUD SQLite a mDB (compuestas)

Todos los enemigos que vuelan que vivan en X región.

En este caso, se ha seleccionado como ejemplo la región “Llanura Estelar”.

```
// 1. Todos los enemigos que vuelan que vivan en X región.  
db.Partidas.find({"LogEnemigos.Enemigos.Region.Nombre": "Llanura Estelar", "LogEnemigos.Enemigos.Vuela":  
    "TRUE"}, {"_id":0, "LogEnemigos.Enemigos.Nombre": 1})
```

La descripción de un ítem cuyo nombre sea X del jugador con id X.

En este caso, obtenemos la descripción del ítem “Seta” que pertenece al jugador con id de partida “game01”.

```
// 2. La descripción de un ítem cuyo nombre sea X del jugador con id X.  
db.Partidas.aggregate([  
    {  
        $match: { "_id": "game01" } // Seleccionar id jugador  
    },  
    {  
        $unwind: "$Items" // Descomponer array  
    },  
    {  
        $match: { "Items.Nombre": "Grandimiel" } // Filtrar por nombre  
    },  
    {  
        $project: {  
            _id: 0,  
            Descripcion: "$Items.Descripcion"  
        }  
    }  
])
```

Nombre de los enemigos que se han enfrentado al jugador con id X.

```
// 3. Mostrar el nombre de los enemigos a los que se ha enfrentado un jugador determinado.  
db.Partidas.aggregate({$match:{"_id":'game02'}},  
{$unwind: '$LogEnemigos'},  
{$unwind: "$LogEnemigos.Enemigos"},  
{$project:{_id:0, Nombre:'$LogEnemigos.Enemigos.Nombre'}});
```

Consultas CRUD SQLite a mDB (agrupadas)

Agrupar las habilidades estrella según su coste y mostrar cuántas habilidades hay de cada coste.

```
// 1. Agrupar las habilidades estrella según su coste y mostrar cuántas habilidades hay de cada coste.
db.Partidas.aggregate([
    {$unwind: "$HabEstrella"},
    {$group:
    {
        _id: "$HabEstrella.CostePE",
        count: { $sum: 1 }
    }
},
])
```

Calcular la media de PC de los personajes según su procedencia.

```
// 2. Calcular la media de PC de los personajes según su procedencia.
db.Partidas.aggregate([{$group:{_id:"$Acompañante.Procedencia.Nombre",MediaPuntosCorazon:{$avg:
"$Acompañante.CurrentPC"}}}]);
```

Calcular el número total de ítems de los jugadores.

```
// 3. Calcular el número total de ítems de los jugadores.
db.Partidas.aggregate([
{
    $project: {
        itemCount: { $size: "$Items" }
    }
},
{
    $group: {
        _id: null,
        totalItems: { $sum: "$itemCount" }
    }
},
])
```

Índices

Jugadores que pueden subir de nivel

```
// Jugadores que pueden subir de nivel
db.Partidas.createIndex({"PuntosNivel":1}, {name:"SubirNivel", partialFilterExpression:{PuntosNivel:{$gt:100}}});
```

Jugadores que pueden abrir la Puerta Milenaria (todas las estrellas)

```
// Reduce el campo de busqueda a los jugadores que pueden abrir la Puerta Milenaria
db.Partidas.createIndex({"CurrentPE":1}, {name:"AbrirPuertaMilenaria",
partialFilterExpression:{CurrentPE:{$lt:8}}});
```

Jugadores que tienen un estado concreto

```
// Jugadores que tienen un estado concreto
db.Partidas.createIndex({"_id":1, "Estado":1});
```

Jugadores que tienen un gran número de puntos flor y de monedas

```
// Jugadores que tienen un gran numero de puntos flor y de monedas
db.Partidas.createIndex({"Monedas": 1, "CurrentPF": 1},{name:"ConsultaSimple1",
partialFilterExpression:{Monedas:{$gt:50}, CurrentPF:{$gt:30}}});
```

Ordenar las Hab. Estrella según su coste

```
// Ordenar habilidades estrella
db.Partidas.createIndex({"HabEstrella.CostePE":1})
```

Ordenar por número de monedas

```
// Ordenar por numero de monedas  
db.Partidas.createIndex({"Monedas":1});
```

Reducir la búsqueda a los enemigos que son débiles

```
// Reduce la busqueda a los enemigos que son debiles  
db.Partidas.createIndex({"LogEnemigos.Enemigos.PuntosAtaque":1, "LogEnemigos.Enemigos.PuntosDefensa":1})
```

Reducir la búsqueda a los enemigos que son boss

```
// Reduce la busqueda a los enemigos que son boss  
db.Partidas.createIndex({"LogEnemigos.Enemigos.Nombre": 1},{name:"EsBoss", partialFilterExpression:{  
LogEnemigos.Enemigos.EsBoss:true}})
```

Evitar nombres duplicados

```
// Evita nombres duplicados  
db.Partidas.createIndex({"LogEnemigos.Enemigos.Nombre":1}, {"unique":true})
```

Mostrar jugadores con menos de 4 Puntos Estrella

```
// Consultas que los puntos estrella sean inferiores a 4  
// Usado en la consulta recurrente 10  
db.Partidas.createIndex({"CurrentPE":1},{partialFilterExpression:{"CurrentPE":{$lt:4}}})
```

Índice compuesto para consultar por procedencia del acompañante

```
// Índice Compuesto para Consultar por Procedencia del Acompanante  
// Para la consulta 14  
db.Partidas.createIndex({"Estado.Acompañante.Procedencia.Nombre":1,"Estado.Acompañante.CurrentPC":1})
```

Índice Compuesto para consultar los valores máximos de PC, PF, PE y PM

```
// Índice Compuesto para consultar los valores máximos de PC, PF, PE y PM  
db.Partidas.createIndex({"MaxPC": 1, "MaxPF": 1, "MaxPE": 1, "MaxPM": 1})
```

Estudio de rendimiento

Al calcular el rendimiento de los índices, no pudimos apreciar gran diferencia aún teniendo 1000 elementos distintos. Debido a la naturaleza de colección única con elementos que contienen una gran cantidad de información y etiquetas `_id` únicas, llegar a un número mucho más elevado habría sido insostenible.

Recursos y bibliografía

https://www.mariowiki.com/Paper_Mario:_The_Thousand-Year_Door_bestiary

- Bestiario de Paper Mario: La Puerta Milenaria (en inglés)

https://www.mariowiki.com/List_of_Paper_Mario:_The_Thousand-Year_Door_enemy_formations

- Formaciones de enemigos en Paper Mario (en inglés), imagen de las batallas.

<https://csvjson.com/csv2json>

- Conversor en línea de archivos CSV a JSON

<https://jsonchecker.com/>

- Validador y formateador de ficheros JSON