ACTIVITAT
Objectius: ■ Processat de dades amb JSON en MongoDB
Instruccions: ■ Resoleu els exercicis proposats i pengeu el repositori a GitHub
Criteris d'avaluació: ■ Cada pregunta té el mateix pes ■ Es valorarà la presentació i els comentaris al codi
Entrega: ■ A Moodle enllaç a repositori GitHub. - A dins del repositori, en un directori anomenat "doc" inclout aquest document completat en format pdf i amb el nom "memoria.pdf"
Repositori d'exemple: ■ https://github.com/jpala4-ieti/DAM-M06-UF03-Punt-Partida-NodeJS

Exercicis

Exercici 1. Mini-Servidor WebSocket de Registre d'Esdeveniments de Joc amb Client Node.js (10 punts)

Crear un servidor WebSocket bàsic en Node.js que escolti connexions entrants. El servidor rebrà missatges en format JSON simulant el moviment 2D d'un jugador en una partida des d'un client Node.js i els emmagatzemarà en una col·lecció de MongoDB.

Tecnologies a utilitzar:

- Node.js
- **Biblioteca `ws`:** Per implementar WebSockets tant al servidor com al client (`npm install ws`).
- MongoDB (utilitzant la mateixa instància de Docker que a la PR3.1)
- Driver de MongoDB (`mongodb`).

Què cal fer?

- Crea una fitxa de requisits per la funcionalitat que es demana resoldre **(2.5 punts)**
- Crea una carpeta anomenada `pr32_websocket_link`
- Crea dos fitxers: `websocket_server.js` i `websocket_client.js` cal implementar aquesta funcionalitat **(5.5 punts)**:
    - El jugador s'ha de moure amb les fletxes.
    - El client enviarà la informació al servidor.
    - Sempre i quan el jugador estigui en moviment la informació de posició es considerarà la mateixa partida. Cada moviment en guardarà a MongoDB (1 document per moviment, però amb informació per associar-lo a la partida).
- Si el jugador passa 10 segons sense moure's es considera finalitzada la partida, el calcula la distància en línia recta entre el punt inicial i final i s'informa al client **(2 punts)**
- En els servidor cal implementar logs (pantalla i fitxer) amb winston.

Fitxa de Requisits: Registre de Moviments de Jugador via WebSocket

Autor: Pablo Vicente Roura
Data: divendres, 25 d'abril 2025

---

1. Objectiu Principal

Desenvolupar una aplicació basada en Node.js formada per un servidor WebSocket i un client que simula el moviment d'un jugador en 2D. El client enviarà comandes de moviment al servidor, que les registrarà en una col·lecció MongoDB. El sistema identificarà quan una partida finalitza (si el jugador deixa de moure's durant 10 segons), calcularà la distància en línia recta entre el punt inicial i final, i retornarà aquesta informació al client.

2. Requisits Funcionals (Què ha de fer)

- **RF-01:** El servidor WebSocket ha d'iniciar-se i escoltar connexions entrants en un port configurable.
- **RF-02:** Els clients WebSocket han de poder connectar-se al servidor.
- **RF-03:** El client ha de detectar les tecles de fletxa i enviar comandes de moviment en format JSON al servidor.
- **RF-04:** El servidor ha de processar les comandes rebudes i registrar cada moviment com un document individual a MongoDB.
- **RF-05:** Cada moviment ha de contenir les coordenades `x`, `y`, un `timestamp`, i un identificador de partida.
- **RF-06:** El servidor ha de mantenir el seguiment d'una partida activa mentre el jugador continuï movent-se.
- **RF-07:** Si passen més de 10 segons sense moviments, el servidor ha de considerar la partida com a finalitzada.
- **RF-08:** Quan es finalitza una partida, el servidor ha de calcular la distància entre la posició inicial i final i enviar un missatge de tipus `gameOver` al client.
- **RF-09:** El servidor ha d'acceptar múltiples partides per client i permetre iniciar-ne de noves automàticament.

3. Requisits No Funcionals (Com ho ha de fer)

- **RNF-01 (Logging):** El servidor ha d'utilitzar la llibreria `winston` per registrar els esdeveniments en consola i en un fitxer (`game_server.log`). Esdeveniments a registrar:
    - Inici del servidor
    - Connexions/desconnexions
    - Moviment rebut
    - Finalització de partida
    - Errors de sistema o missatge invàlid
- **RNF-02 (Persistència):** La base de dades ha de ser MongoDB, utilitzada amb el driver oficial (`mongodb`). Les dades de connexió han de ser configurables mitjançant variables d'entorn.
- **RNF-03 (Robustesa):** El servidor ha de gestionar errors comuns (com errors de connexió amb MongoDB o JSON mal format) i continuar funcionant si és possible, registrant l'error.
- **RNF-04 (Estructura del codi):** El projecte ha d'estar organitzat dins una carpeta `pr32_websocket_link` i contindrà els fitxers `websocket_server.js` i `websocket_client.js`.

4. Format de Missatges JSON

Missatge amb la posició inicial

```
case 'initialState':
    console.log('📦 Estat inicial rebut.');
    currentPosition.x = data.x;
    currentPosition.y = data.y;
```

Missatge amb l'actualització de la nova posició

```
case 'positionUpdate':
    currentPosition.x = data.x;
    currentPosition.y = data.y;
    break;
```

Missatge de quan el game s'ha acabat

```
case 'gameOver':
    console.log(`\n🏁 PARTIDA FINALITZADA (ID: ${data.gameId})`);
    console.log(`📏 Distància recorreguda: ${data.distance}`);
    console.log(`⏱️ Durada: ${new Date(data.startTime).toLocaleTimeString()} - ${new Date(data.endTime).toLocaleTimeString()}`);
    console.log('🔁 Mou-te per iniciar una nova partida.');
    break;
```

Missatge de error en el servidor

```
case 'error':
    console.warn(`⚠️ Error del servidor: ${data.message}`);
    break;
```