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.
- mirar formato de los JSON!

Annex

Exemple fitxa de requisits

Sistema de Monitorització de Sensors en Temps Real:

Crea un servidor Node.js que utilitzi WebSockets per rebre lectures simulades de sensors (per exemple, temperatura i humitat) en format JSON des de múltiples clients Node.js. Emmagatzema cada lectura rebuda, juntament amb un timestamp, en una col·lecció de MongoDB. El servidor podria implementar una lògica per detectar lectures anòmales (fora d'un rang esperat) i notificar-ho als clients o registrar-ho de manera especial.

Fitxa de Requisits: Sistema de Monitorització de Sensors

Autor: [El Teu Nom o Equip] **Data:** [Data de Creació, ex: 2025-03-27]

1. Objectiu Principal

Desenvolupar un prototip funcional amb Node.js que actuï com a servidor WebSocket per rebre lectures simulades de diversos sensors (com temperatura i humitat) des de clients Node.js. Les lectures rebudes s'han d'emmagatzemar en una base de dades MongoDB juntament amb un timestamp, i el sistema ha de poder identificar i registrar lectures que es considerin anòmales (fora d'un rang predefinit).

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 (simulats) han de poder connectar-se al servidor.
- **RF-03:** Els clients han d'enviar lectures de sensors periòdicament en format JSON.
- **RF-04:** El format JSON de les lectures ha d'incloure `sensorId` (string), `type` (string, ex: "temperature"), i `value` (number).
- **RF-05:** El servidor ha de rebre i processar correctament els missatges JSON dels clients.
- **RF-06:** El servidor ha de validar l'estructura bàsica dels missatges rebuts. Els invàlids s'han de descartar i registrar (log).
- **RF-07:** El servidor ha d'afegir un `timestamp` (data i hora UTC) a cada lectura vàlida rebuda.

- **RF-08:** El servidor ha d'emmagatzemar cada lectura vàlida (amb `sensorId`, `type`, `value`, `timestamp`) com un document individual en una col·lecció de MongoDB.
- **RF-09:** El sistema ha de permetre configurar rangs de valors considerats normals per a cada tipus de sensor (ex: Temperatura 0-50°C).
- **RF-10:** El servidor ha de comprovar si el `value` de cada lectura rebuda cau dins del rang normal definit per al seu `type`.
- **RF-11:** El servidor ha de registrar de forma específica (log) qualsevol lectura detectada com a anomalia (fora de rang).
- **RF-12:** El servidor ha de gestionar les desconnexions dels clients i registrar l'esdeveniment.

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

- **RNF-01 (Logging):** El servidor ha d'implementar logs amb `winston`. S'han de registrar els esdeveniments clau (inici, connexió/desconnexió, error, dada rebuda, dada emmagatzemada, anomalia detectada) a la consola i a un fitxer (`sensor_server.log`).
- **RNF-02 (Mantenibilitat):** El codi ha d'estar comentat. La configuració essencial (port del servidor, URI de connexió a MongoDB, rangs d'anomalia) ha de ser fàcilment modificable (p. ex., variables d'entorn o fitxer de configuració).
- **RNF-03 (Fiabilitat Bàsica):** El servidor hauria de poder gestionar errors comuns (ex: error de connexió a BD) registrant l'error sense aturar-se completament, si és possible.

4. Format Missatge JSON (Client -> Servidor)

Els clients enviaran lectures amb l'estructura següent:

```
{
  "sensorId": "string", // Identificador únic del sensor (ex: "TEMP-001", "HUM-A")
  "type": "string",     // Tipus de mesura (ex: "temperature", "humidity")
  "value": number       // Valor numèric de la mesura (ex: 23.5, 45.2)
}
```
