Fitxa de Requisits: Sistema de Monitorització de Sensors

Autor: Mario Pes
Data de Creació: 07/06/2026

---

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 els següents camps:

- `sensorId` (string): Identificador únic del sensor.
- `type` (string): Tipus de mesura (ex: "temperature", "humidity").
- `value` (número): Valor numèric de la mesura (ex: 23.5, 45.2).

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 missatges 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 tipus de sensor.

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 (Servidor -> Client)

El servidor envia missatges al client en el següent format:

1. **Posició del Jugador (després de moure-se):**

Quan el jugador es mou i es vol enviar la nova posició, el servidor envia un missatge de tipus **"position"** amb les coordenades actualitzades `x` i `y`:

```
{
  "type": "position",   // Tipus de missatge (posició del jugador)
  "x": 10,          // Nova coordenada X
  "y": 20           // Nova coordenada Y
}
```

2. **Partida Finalitzada (quan el jugador ha estat inactiu més de 10 segons):**

Quan el jugador està inactiu durant més de 10 segons, el servidor finalitza la partida, calcula la distància entre la posició inicial i final, i envia un missatge de tipus **"game-over"** amb la distància recorreguda:

```
{
  "type": "game-over",  // Tipus de missatge (fi de la partida)
  "distance": 14.14       // Distància recorreguda pel jugador (en aquest cas, per exemple, 14.14 unitats)
}
```