Esercizio: Middleware Detective

Obiettivo

Capire come le richieste HTTP attraversano una **catena di middleware**, ognuno con un compito specifico (log, timer, trasformazioni).
L'obiettivo è visualizzare nel frontend il risultato del passaggio attraverso questi livelli, come una **catena di montaggio** che elabora la richiesta passo dopo passo.

Introduzione teorica

In Express, un **middleware** è una funzione che riceve tre parametri:

- `req` → rappresenta la richiesta del client (dati, header, parametri)
- `res` → rappresenta la risposta che il server invierà
- `next()` → serve per **passare al middleware successivo**

Ogni middleware può:

- leggere o modificare la richiesta (`req`);
- aggiungere log o informazioni;
- validare dati o bloccare l'accesso;
- oppure interrompere il flusso se trova un errore.

Schema del flusso di una richiesta

```
Richiesta → [Logger] → [Timer] → [Uppercase] → [Controller finale] → Risposta
```

- **Logger:** registra chi e quando effettua la richiesta
- **Timer:** misura il tempo di elaborazione
- **Uppercase:** modifica i dati prima della risposta
- **Controller finale:** invia il risultato al client

Struttura del progetto

```
middleware-lab/
├── server.js              ← file principale del server Express
├── middleware/            ← cartella con i middleware personalizzati
│   ├── logger.js
│   ├── timer.js
│   └── uppercase.js
└── public/                ← frontend statico con HTML e JS
    ├── index.html
    └── script.js
```

Passaggi operativi

1️⃣ Crea la cartella del progetto e inizializza npm.
2️⃣ Installa **Express** come unica dipendenza.
3️⃣ Crea i tre middleware `logger`, `timer` e `uppercase` (ognuno con una sola responsabilità).
4️⃣ Crea il **server Express** che usa i middleware e serve un piccolo frontend.
5️⃣ Nel **frontend**, aggiungi un form per inviare un messaggio al server e visualizzare la risposta JSON.
6️⃣ Osserva nel terminale l'ordine in cui i middleware vengono eseguiti e i log generati.

Spunti per il codice

- Il **logger** deve stampare metodo e rotta di ogni richiesta.
- Il **timer** deve misurare quanto tempo impiega il server per rispondere.
- Il middleware **uppercase** deve modificare il testo ricevuto prima di inviarlo indietro.
- Il **frontend** deve inviare una richiesta POST e mostrare la risposta trasformata.

Obiettivo finale

Quando invii un messaggio dal frontend, il server lo elabora attraverso i middleware e restituisce una risposta modificata.
Nel terminale del server compariranno log ordinati che mostrano **il percorso della richiesta**.

In questo modo, puoi **visualizzare in pratica** come funziona il concetto di *"flusso di middleware"* in Express.