Server Musei via openAPI (CRUD + REST)

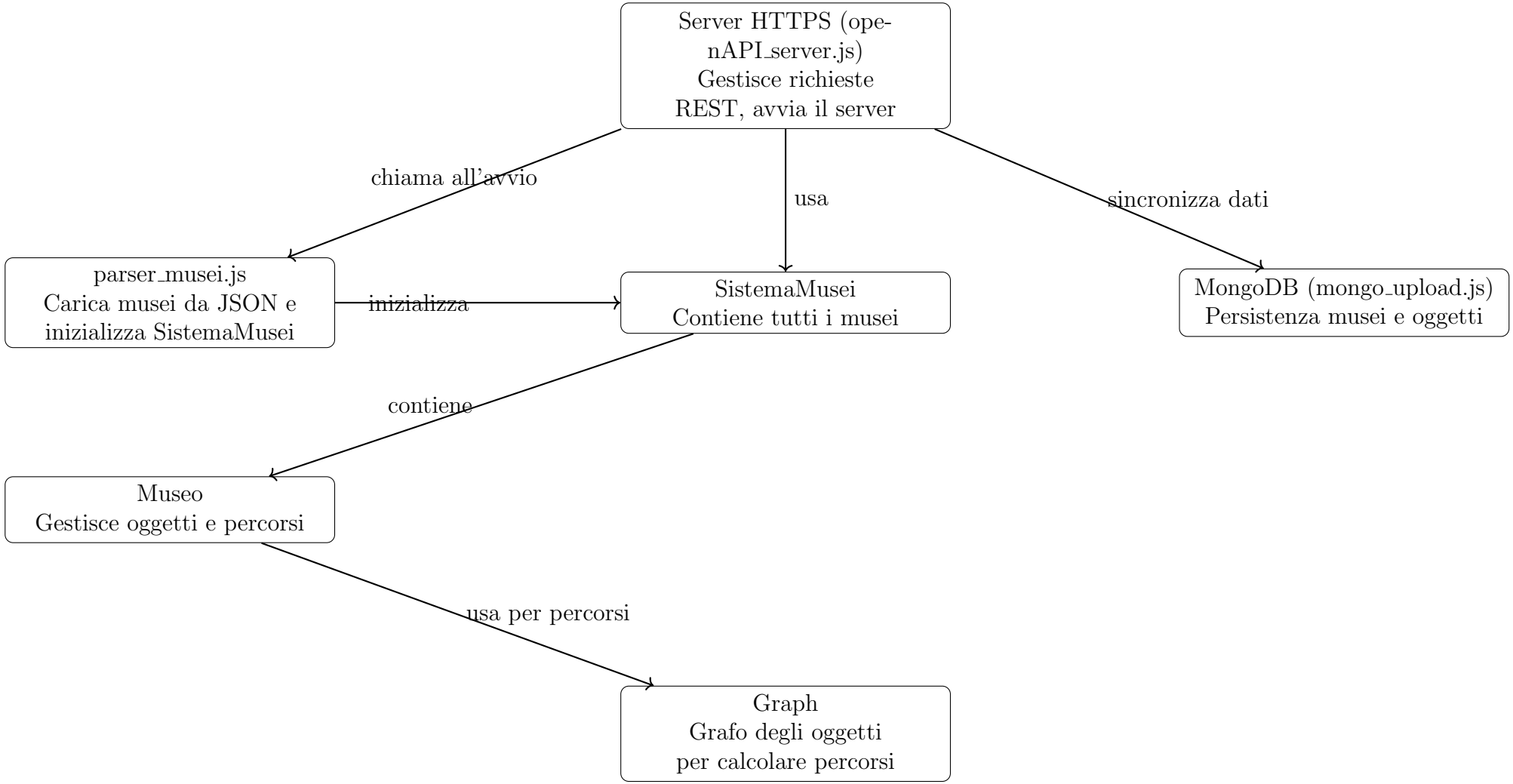
Strutture del codice

December 3, 2025

1. Panoramica delle strutture principali

- **Server HTTPS (openAPI_server.js)**: punto di ingresso del sistema, gestisce le richieste HTTP.
- **SistemaMusei (sistema_musei.js)**: struttura che contiene tutti i musei.
- **Museo (museo.js)**: rappresenta un singolo museo e gestisce i suoi oggetti.
- **Graph (graph.js)**: struttura a grafo per la connessione tra oggetti all'interno di un museo.
- **MongoDB (mongo_upload.js)**: persistenza dei dati, sincronizza musei e oggetti su database.
- **parser_musei.js**: carica musei da file JSON iniziale.

2. Connessioni tra le strutture



3. Funzionamento ad alto livello

1. Il **server HTTPS** riceve richieste API.
2. Viene usato **SistemaMusei** per ottenere o modificare dati dei musei.
3. Ogni **Museo** gestisce i suoi **oggetti** e usa il **Graph** per calcolare percorsi tra oggetti.
4. Tutte le modifiche vengono salvate su file JSON e sincronizzate con **MongoDB**.
5. All'avvio, **parser_musei.js** carica il JSON e inizializza **SistemaMusei**.

4. Note aggiuntive

- Middleware sicurezza gestisce API key e CORS.
- Logging delle richieste per debug.
- HTTPS con certificati TLS, opzionale solo localhost.

5. Sicurezza e protezione

- **API Key**: tutte le richieste devono includere un header `x-api-key` valido per essere autorizzate.
- **Connessione limitata a localhost**: il server accetta richieste solo da `localhost` (127.0.0.1).
- **TLS / HTTPS**: tutte le comunicazioni avvengono tramite HTTPS per garantire cifratura dei dati in transito.
- **Autenticazione**: il server verifica la chiave API prima di processare qualsiasi richiesta REST.
- **Uso tipico**: in sviluppo, i certificati TLS possono essere self-signed; in produzione sarebbe necessario un certificato valido.