Server SVG Musei (Python + Flask)

Architettura e funzionamento

December 16, 2025

1. Panoramica generale

Il **Server SVG Musei** è un servizio Python basato su **Flask** che ha il compito di:

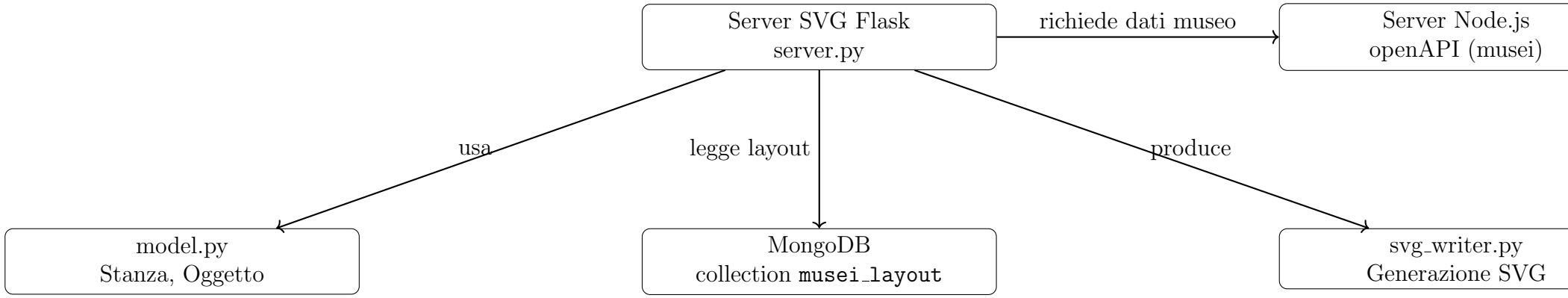
- Generare dinamicamente **mappe SVG dei musei**.
- Recuperare la **struttura spaziale (layout)** da **MongoDB**.
- Recuperare i **dati logici del museo** (oggetti, connessioni) dal server Node.js.
- Combinare layout e dati logici per produrre una rappresentazione grafica coerente.

Il server non modifica dati: ha un ruolo **read-only e di rendering**.

2. Componenti principali

- **server.py (Flask)**: punto di ingresso HTTP, espone endpoint per ottenere SVG.
- **MongoDB (collection `musei_layout`)**: contiene i layout delle stanze per ciascun museo.
- **Server Node.js (openAPI)**: fornisce i dati logici del museo via REST.
- **model.py**: definisce le classi `Stanza` e `Oggetto`.
- **layout.py**: calcola corridoi e posizionamento spaziale delle stanze.
- **svg_writer.py**: converte strutture interne in SVG finale.

3. Connessioni tra i componenti



4. Endpoint esposti

Il server espone endpoint HTTP di sola lettura:

- `/NomeMuseo`
- `/NomeMuseo/edge_mode`
- `/NomeMuseo/edge_mode/f1/f2`

Risposta: un documento `image/svg+xml` che rappresenta la mappa del museo.

5. Flusso di funzionamento

1. Il client richiede l'SVG di un museo.
2. Il server Flask attende che il server Node sia online.
3. Il layout del museo viene recuperato da MongoDB.
4. I dati logici (oggetti, connessioni) vengono richiesti al server Node.

5. Vengono istanziate le classi `Stanza` e `Oggetto`.
6. Il layout viene calcolato (stanze + corridoi).
7. Il risultato finale viene convertito in SVG.

6. Dati gestiti

6.1 Layout (MongoDB)

Ogni documento layout contiene:

- Identificatore del museo (`_id`).
- Griglia di stanze con coordinate (`row`, `col`).
- Tipo della stanza (ingresso, uscita, bagno, servizio).

6.2 Dati museo (Node.js)

Il server Node fornisce:

- Nome e città del museo.
- Lista degli oggetti.
- Connessioni logiche tra oggetti.

7. Sicurezza e robustezza

- **API Key**: richiesta per comunicare con il server Node.
- **Timeout controllati**: evita blocchi su servizi non disponibili.
- **Attesa attiva all'avvio**: Flask parte solo quando Node è pronto.
- **HTTPS supportato**: comunicazione cifrata con il server Node.

8. Ruolo nel sistema complessivo

Il server Python:

- Non modifica dati persistenti.
- Non gestisce CRUD.
- È responsabile esclusivamente della **visualizzazione**.
- Separa nettamente **logica dei dati** e **rappresentazione grafica**.