Progetto IUM-TWEB
(12 CFU)

Elena Derosas
1049692

27 agosto 2025

Introduzione

Il sistema implementa un'architettura a microservizi per consentire a fan e giornalisti di esplorare dati sui film. È stato realizzato un Main Server in Express che espone un'API unificata e una pagina HTML+JS+CSS+Handlebars; i dati statici sono serviti da un microservizio Spring Boot con Postgres, mentre i dati dinamici sono gestiti da un microservizio Express con MongoDB. La progettazione mantiene il Main Server "leggero", demandando le elaborazioni al livello dati, e integra Socket.IO per la chat in tempo reale, in linea con i requisiti.

1 Task Tecnici

1.1 Task 1 — Main Server

Soluzione

Il Main Server in Express funge da unico punto d'accesso per il frontend e orchestra le richieste verso i microservizi. La documentazione è esposta tramite OpenAPI/Swagger; vengono applicate normalizzazioni minime ai payload e vengono gestiti in modo coerente CORS, timeout e logging. I template Handlebars sono renderizzati lato server nel Main Server, semplificando l'integrazione con le API.

Issues

L'integrazione ha messo in luce disallineamenti di CORS/timeout tra servizi e una gestione eterogenea degli errori Axios; la situazione è stata stabilizzata con origini e timeouts condivisi e con una mappatura coerente degli errori lato Main Server.

Requisiti

Il Main Server è stato mantenuto deliberatamente leggero: proxy, validazioni non onerose e minime trasformazioni dei dati; espone inoltre la pagina Handlebars e la documentazione Swagger richieste.
Le chiamate ai microservizi avvengono tramite Axios incapsulato in un modulo di proxy: sono configurati gli endpoint base (Spring Boot e Other Express), i timeout e un set di metodi ammessi; gli errori vengono arricchiti e poi inoltrati al middleware di gestione.

Limitazioni

Non sono presenti circuit breaker, retry con backoff o caching; tali funzionalità potranno essere introdotte senza modifiche alle API esterne qualora i volumi lo richiedano.

1.2 Task 2 — Spring Boot + Postgres

Soluzione

I dati statici risiedono in Postgres ed è Spring Boot a esporli tramite endpoint per ricerca, suggerimenti, dettaglio e statistiche globali. L'interfaccia REST è disaccoppiata dal modello persistente usando DTO: in ingresso i parametri sono mappati su `MovieFilterRequestDto` con validazioni di range; in uscita si impiegano `ApiResponse<T>` e `PagedApiResponse<List<MovieSummaryDto>>` con `PaginationMetadataDto`, `SuggestionsResponse` per i suggerimenti e `MovieDetailDto` per i dettagli. In questo modo validazioni, paginazione e forma del payload sono gestite nel servizio Java, mentre il Main Server si limita all'orchestrazione e a minime normalizzazioni.

Per l'accesso ai dati è utilizzato Spring Data JPA (Hibernate) come ORM su Postgres: repository tipizzate, paginazione con `Page`, conteggi globali e mapping verso DTO garantiscono un contratto REST stabile e separato dalle entità.

Issues

Le API dei servizi presentavano differenze di naming e formattazione dei payload, e spesso si generavano risposte incoerenti. Si è risolto con l'introduzione di un mapping unico nel Main Server.

Requisiti

Come richiesto, l'elaborazione dei dati avviene nel servizio Spring Boot (Postgres) tramite query e DTO validati, mentre il Main Server si limita all'instradamento e a minime normalizzazioni.

Limitazioni

Non è stato abilitato un meccanismo di caching dei risultati, utile per ricerche ripetute.

1.3 Task 3 — Express + MongoDB

Soluzione

Il microservizio Express con MongoDB gestisce recensioni e statistiche. L'accesso ai dati avviene tramite Mongoose (ODM), con schemi e modelli per `reviews`, `messages` e `rooms`; sono definiti indici composti per le query più frequenti e metodi statici per paginazione, recuperi temporali e aggregazioni. Le pipeline di aggregazione calcolano medie, valori estremi, distribuzioni fresh/rotten, incidenza dei top critics e relative percentuali. Gli endpoint espongono tali risultati e il Main Server li integra, quando necessario, con i dati statici di Spring Boot in un'unica risposta coerente per il client.

Issues

La qualità eterogenea dei dati (date come stringhe, punteggi mancanti) ha richiesto filtri accurati in aggregazione; sono stati impostati vincoli minimi sui range numerici e indici composti per evitare letture sequenziali dell'intera collezione.

Requisiti

La collocazione delle recensioni su MongoDB e l'esposizione tramite un servizio Express separato rispettano la divisione dinamico/statico; il Main Server intermedia senza eseguire calcoli pesanti, limitandosi all'orchestrazione e alla composizione delle risposte.

Limitazioni

Le pipeline non sono assistite da cache: su dataset molto grandi una cache applicativa può migliorare i tempi di risposta.

1.4 Task 4 — Chat in Tempo Reale (Socket.IO)

Soluzione

Socket.IO è integrato nel Main Server per mantenere un'unica origine e semplificare CORS e handshake. Il sistema di live chatting è organizzato in room tematiche.

Issues

Lo studio e la configurazione di Socket.IO (CORS, trasporti, path, client/server) hanno richiesto più tempo del previsto. Anche la sua implementazione corretta ha richiesto più di un tentativo.

Requisiti

L'uso di Socket.IO soddisfa l'obbligo di chat real-time per IUM-TWEB; l'integrazione nel Main Server preserva la leggerezza dell'entrypoint, dato che lo scambio eventi è I/O-bound e la persistenza resta di competenza esterna.

Limitazioni

La persistenza dei messaggi nelle diverse room è stata prevista nel servizio Mongo tramite API REST (sono presenti modelli per messaggi e stanze), ma l'integrazione con il frontend non è stata completata per mancanza di tempo. La possibilità resta valida e implementabile in un secondo momento.

1.5 Task 5 — Frontend

Soluzione

L'interfaccia è servita dal Main Server e consente di cercare ed esplorare i dati. Le richieste di ricerca e suggerimenti vanno a Spring Boot; i dettagli dei film sono arricchiti con recensioni e statistiche dal servizio Mongo. La chat è accessibile via Socket.IO. Le viste sono generate con Handlebars a partire dalle risposte normalizzate dal Main Server, utilizzando Bootstrap 5 e Bootstrap Icons per lo stile. Inoltre, per le scelte di styling delle pagine (palette, tipografia, spaziature e componenti) ci si è avvalsi del supporto di un assistente AI, validando manualmente le proposte.

Issues

È stato necessario uniformare il formato dei dati provenienti dai servizi e gestire casi in cui una fonte non era momentaneamente disponibile. È stata utilizzata l'AI per rendere la vista piacevole e ordinata.

Requisiti

La pagina HTML+JS+CSS+Handlebars soddisfa la richiesta di esplorazione dei dati e sfrutta il Main Server solo per instradamento e normalizzazioni leggere.

Limitazioni

Non sono stati utilizzati framework SPA né cache client avanzate; se servisse più interattività si potrà evolvere la UI mantenendo invariato il contratto con il Main Server.

Conclusioni

L'architettura soddisfa i requisiti principali (Main Server leggero, separazione tra dati statici e dinamici, chat real-time), ma il codice risulta ancora acerbo nella scrittura e merita maggiore ordine: coerenza nei nomi, modularizzazione più chiara, gestione degli errori uniforme e riduzione delle duplicazioni.

La gestione delle dipendenze è stata affidata a Maven per il servizio Spring Boot (pom.xml) e a Node/npm per il Main Server e il servizio Express+Mongo (package.json, con `package-lock.json`). La configurazione è esternalizzata via variabili d'ambiente (.env) e `application.properties`, così da separare codice e ambienti di esecuzione.
La documentazione è stata redatta, ma non se ne garantisce la completezza e restano dubbi sulla correttezza formale rispetto agli standard.

Divisione del Lavoro

Il lavoro è stato svolto integralmente dalla sottoscritta, non facendo parte di un gruppo.

Extra Information

Si consiglia la lettura del README.

Il sistema richiede Postgres e MongoDB attivi.

Setup dei database È necessario un setup iniziale dei database: i CSV sono stati puliti (pre-processing) prima dell'import. Sono previste due modalità:

1. *Setup completo*: pulizia dei csv e input dei dati processati in Postgres e in MongoDB. In tal caso è necessario creare un proprio file `solution/database/.env` sulla base del template presente e l'installazione delle dipendenze Python (`solution/database/requirements.txt`);
2. *Ripristino rapido*: utilizzo dei backup già pronti in `solution/database/backups.zip`, da estrarre e ripristinare su MongoDB (ad es. tramite import JSON) e su Postgres (ad es. `psql` su dump SQL).

Environment Per impostazione predefinita:

- il Main Server è su `MAIN_SERVER_PORT=3000` con Swagger su `/api-docs`;
- Spring Boot risponde su `SPRING_BOOT_SERVER_URL=http://localhost:8081`;
- il servizio Mongo su `OTHER_EXPRESS_SERVER_URL=http://localhost:3001`.

È necessario creare i file `solution/main-server/.env` e `solution/express-mongo-server/.env` sulla base dei template presenti, con le variabili di ambiente personali, comprese quelle per la comunicazione con i database; Per il server Spring Boot, le variabili sono accessibili e personalizzabili tramite `solution/springboot-server/src/main/resources/application.properties`.

Bibliografia

Riferimenti bibliografici

[1] Express.js Documentation. https://expressjs.com/

[2] Spring Boot Reference Guide. https://docs.spring.io/spring-boot/docs/current/reference/html/

[3] Spring Data JPA Reference. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

[4] PostgreSQL Documentation. https://www.postgresql.org/docs/

[5] MongoDB Manual. https://www.mongodb.com/docs/

[6] MongoDB Aggregation Pipeline. https://www.mongodb.com/docs/manual/aggregation/

[7] Mongoose Docs. https://mongoosejs.com/docs/

[8] Socket.IO Documentation. https://socket.io/docs/v4/

[9] OpenAPI 3.0.3 Specification. https://spec.openapis.org/oas/v3.0.3

[10] swagger-ui-express (npm). https://www.npmjs.com/package/swagger-ui-express

[11] Handlebars.js Guide. https://handlebarsjs.com/guide/

[12] Bootstrap 5 Docs. https://getbootstrap.com/docs/5.3/getting-started/introduction/

[13] Bootstrap Icons. https://icons.getbootstrap.com/

[14] Axios Documentation. https://axios-http.com/