Assignment #4 - Digging into RESTful Architectures

Giacomo Totaro
giacomo.totaro2@studio.unibo.it

Settembre 2024

1 Report

L'architettura RESTful, come descritta nell'articolo "Principled Design of the Modern Web Architecture" di Roy Fielding, rappresenta un paradigma robusto e scalabile per costruire sistemi distribuiti. REST ha dimostrato la sua efficacia in vari contesti, soprattutto nel Web, grazie ai suoi principi di interfaccia uniforme, comunicazione senza stato e utilizzo di risorse ben definite. Tuttavia, con l'avvento delle architetture basate su microservizi, ci si chiede se e come REST possa essere sfruttato, o addirittura esteso, per affrontare le sfide di scalabilità, resilienza e distribuzione di tali sistemi.

1.1 REST come base solida per i microservizi

I microservizi sono piccole unità autonome che possono essere sviluppate, distribuite e scalate indipendentemente. Uno dei pilastri dell'architettura a microservizi è la separazione delle responsabilità. Da questo punto di vista, REST, con il suo modello client-server e il concetto di risorse ben definite tramite URL, fornisce una struttura naturale per implementare microservizi che interagiscono tra loro attraverso API chiare e autodocumentante.
Un altro principio chiave di REST è la comunicazione stateless: ogni richiesta HTTP deve contenere tutte le informazioni necessarie per il server a elaborarla, senza bisogno di mantenere lo stato tra una richiesta e l'altra. Questo principio si allinea perfettamente con l'idea di scalabilità dinamica nei microservizi, poiché un'istanza di microservizio può essere replicata e distribuita senza dover mantenere sessioni di stato condivise. Grazie a questa caratteristica, REST può supportare un'elevata scalabilità orizzontale, consentendo ai microservizi di gestire volumi elevati di traffico semplicemente aggiungendo ulteriori istanze.

1.2 Limiti di REST nelle architetture a microservizi

Nonostante i suoi vantaggi, REST presenta alcuni limiti intrinseci quando applicato a sistemi di microservizi distribuiti altamente complessi. Uno dei principali limiti è la mancanza di supporto nativo per la comunicazione asincrona. In contesti moderni, i microservizi spesso comunicano tra loro in modo asincrono per ridurre i tempi di attesa e migliorare la resilienza. REST, basato su HTTP, è orientato alla comunicazione sincrona, in cui il client attende una risposta immediata dal server.
Un altro aspetto critico riguarda la gestione delle transazioni distribuite. In sistemi complessi di microservizi, è frequente la necessità di coordinare transazioni tra più servizi. REST, essendo pensato per operazioni atomiche e indipendenti, non gestisce bene scenari che richiedono consistenza distribuita. Strumenti come il protocollo Saga vengono spesso implementati in microservizi per gestire transazioni distribuite in maniera più resiliente, qualcosa che REST non supporta direttamente.
Quindi, essendo orientato a operazioni sincrone tramite HTTP, REST non gestisce bene la necessità di coordinare più servizi in contesti distribuiti complessi. Per questo motivo, viene spesso esteso o affiancato da altri modelli come le architetture **event-driven**, che sfruttano eventi e strumenti come Kafka per la comunicazione asincrona, o da soluzioni ibride con GraphQL per query più flessibili. REST rimane efficace, ma può necessitare di adattamenti per rispondere alle esigenze moderne.

1.3 Conclusioni

REST è una solida base per la progettazione di architetture a microservizi distribuiti grazie ai suoi principi di statelessness, separazione client-server e uso di risorse ben definite. Tuttavia, per affrontare le sfide moderne della comunicazione asincrona e della scalabilità estrema, potrebbe essere necessario estendere o integrare REST con altri modelli e tecnologie. Infine, un'ulteriore riflessione riguarda l'uso di REST in architetture multi-protocollo, dove REST è solo uno dei metodi di comunicazione tra i microservizi. In questi scenari, si potrebbe utilizzare REST per l'accesso alle API pubbliche o per operazioni CRUD, mentre tecnologie più avanzate come gRPC o sistemi di messaggistica asincrona vengono impiegate per la comunicazione interna ad alte prestazioni. Questo approccio potrebbe combinare i punti di forza di REST con soluzioni che gestiscono meglio la comunicazione asincrona e la consistenza distribuita.