GraphQL

Creato da Facebook nel 2012

[GraphQL](https://graphql.org/) è un meccanismo di accesso alle API creato da Facebook nel 2012, ma rilasciato pubblicamente solo nel 2015. Si tratta di un **linguaggio di interrogazione** (o **query language**) nato con lo scopo di offrire un’alternativa ai classici schemi, ed in particolare di superare i limiti ravvisati nel paradigma [REST](https://www.html.it/guide/restful-web-services-la-guida/).

GraphQL è un’estensione di REST

GraphQL non rappresenta una rottura totale con REST, bensì prosegue sulla stessa via ereditandone i principali elementi fondanti:

* fa leva sul protocollo **HTTP** sfruttando gli elementi che al suo interno lo predispongono naturalmente ad essere un vettore di richieste e risposte tra client e server (metodi, intestazioni, parametri, etc.);
* comporta un **basso overhead** risultando pertanto particolarmente efficiente;
* si ispira al **principio della semplicità** non richiedendo strutture dati o meccanismi particolarmente complessi per svolgere i compiti assegnati.

Quello che per GraphQL può essere considerato però il principio cardine – aspetto in cui vuole contraddistinguersi da REST – è la **capacità di permettere ai client di richiedere solo i dati cui si è interessati**, con la massima precisione. Questo sembra essere un obiettivo troppo spesso mancato da REST, che è sempre stato afflitto da due problemi. Il primo è l’**over-fetching**, situazione in cui un endpoint risponde troppo genericamente offrendo una gran mole di dati che il client in buona parte non usa, ma che deve comunque elaborare. Il secondo problema, opposto al primo, prende nome di **under-fetching** per cui un endpoint risponde con informazioni troppo stringate per cui un client può avere un set completo di dati solo inoltrando più richieste con le immaginabili conseguenze in termini di efficienza e consumo di banda, tanto per sè stesso quanto per il server.

Oltre a questo aspetto, GraphQL presenta ulteriori punti di forza che ne hanno permesso un veloce successo tra un grande pubblico di sviluppatori:

* mentre REST si fonda sul principio della “mappatura” di risorse con endpoint di rete, GraphQL permette di accedere a più risorse attraverso una singola richiesta. Questo permette di **evitare richieste multiple** a più URL, come può capitare in REST: una caratteristica introdotta strizzando l’occhio al mondo mobile;
* si favorisce al massimo l’utilizzo dei **tipi di dato** per indurre una maggiore identificazione dei dati ed una migliore visibilità sugli errori;
* la flessibilità nella formulazione delle query permette di predisporre ulteriori campi senza compromettere la correttezza delle interrogazioni già in produzione. Questo è sempre stato un limite di REST nel quale, infatti, si ricorre spesso al *versioning* delle funzionalità per poter distinguere con quali API un client può dialogare. **Il versioning non trova pertanto applicazione in GraphQL**.

Slide 3

Una cosa interessante da sottolineare è che GraphQL non è legato a nessun database specifico (o motore di archiviazione per quella materia) ed è invece supportato da codice esistente. Ciò significa che, a differenza delle API REST (in cui il client interagisce prima con codice arbitrario scritto dai programmatori e questo codice raggiunge il database); il client interagisce prima con GraphQL, che a sua volta interagisce con codice arbitrario e alla fine finisce di parlare con il database. Un diagramma più utile per rappresentare questa situazione è questo:

Perché Mock?

Mock i dati che un backend restituirebbe è molto utile per due motivi principali: Ti consente di iniziare a sviluppare una funzionalità di frontend quando non disponi ancora di un backend funzionante. Questo è fondamentale per i progetti in cui i componenti frontend e backend sono spesso sviluppati in parallelo. Ti consente di eseguire test localmente senza connetterti a un vero backend, che è molto più veloce e sicuro. Man mano che la tua base di codice cresce e la tua app diventa più complessa, l'avvio di tutta l'infrastruttura server solo per eseguire alcuni test non è fattibile. Se Mock la tua API di backend ha vantaggi così evidenti, perché non lo fanno tutti? Penso che sia perché Mock spesso sembra troppo disturbo per valerne la pena.