Quando è giusto usare Node.js per un progetto web

In questo articolo risponderemo ad una domanda fondamentale su Node.js e sul suo impiego in un progetto web.

Quando è giusto usare Node.js?

Il primo aspetto da chiarire potrà risultare sorprendente per molti: Node.js non è stato concepito in origine come environment orientato al web. Infatti, Node.js ha solo alcuni moduli core che gestiscono i protocolli alla base del web (HTTP in primis) che consentono solo le operazioni fondamentali relative allo sviluppo di un progetto web.

Node.js, di fatto, è agnostico riguardo all'ambito di applicazione finale delle sue funzionalità: può essere usato sia per gestire embedded system che per sistemi industriali che per sviluppare progetti web.

PHP, al contrario, è un ottimo esempio di linguaggio concepito quasi esclusivamente per il web. Prendiamo ad esempio la funzione nl2br(): questa funzione trasforma le interruzioni di riga in elementi HTML `br`. Notiamo come l'output predefinito di PHP sia appunto il formato HTML, ossia il formato alla base di tutte le pagine web. PHP nasce nel 1994 e fin dall'inizio il suo design è stato sempre orientato al web. Node.js nasce nel 2008 e il suo design è sempre stato generico e multipurpose.

PHP dispone di centinaia di funzioni e classi native che coprono un'ampia gamma di caratteristiche di un progetto web; Node.js utilizza invece i moduli NPM per aggiungere tutte quelle caratteristiche che servono allo sviluppo web. Ad esempio ExpressJS estende il modulo core `http` aggiungendo le funzionalità di routing, middleware e di gestione delle richieste e risposte HTTP. In PHP un framework che segue un approccio simile è Laravel.

JavaScript è un linguaggio estremamente flessibile: non ha bisogno di avere funzioni built-in così numerose come PHP perché semplicemente si possono implementare sfruttando le caratteristiche base del linguaggio e dello standard ECMAScript.

Rispetto a PHP, Node.js non ha l'enorme diffusione presso gli hosting provider che invece supportano lo stack LAMP o LEMP. Infatti l'installazione di Node.js su un server richiede alcuni accorgimenti preliminari che non sono necessari quando si installa invece PHP, MySql e il web server Apache (o nginx). Heroku è un ottimo esempio di un hosting condiviso per applicazioni in Node.js, ma cercare di replicare la sua complessa gestione delle immagini virtuali delle applicazioni e del networking in uso richiede delle competenze sistemistiche che la maggior parte degli sviluppatori non è in grado di possedere in quanto è richiesto un notevole know-how ed esperienza sul campo.

Heroku ha inoltre uno svantaggio in termini di costi: aggiungere ad esempio l'istanza di un database come MongoDB può comportare spesso una spesa maggiore del costo mensile di un istanza VPS autonoma.

Node.js è l'opzione giusta quando si vogliono avere **performance**, **scalabilità** e **adattabilità nel futuro**. Il consumo di risorse di un'istanza Node, che è single-threaded by design, è molto basso e contenuto. PHP può avvicinarsi a questi risultati solo a patto di implementare un sistema di caching multi-livello unito ad un setup specifico, mentre Node è già pronto alla produzione quasi sempre con ottimizzazioni minime.

Node.js non è l'opzione giusta quando la data di consegna di un progetto è ravvicinata. Se infatti dobbiamo sviluppare un e-commerce completo di backend e frontend in 2-3 settimane, è da tenere presente che con Node il tempo richiesto è in media di 30-45 giorni a meno di non usare una soluzione che permetta di abbattere i tempi dello sviluppo del backend, che è in assoluto la parte più complessa soprattutto per quello che riguarda la user experience. In PHP invece per il backend esistono già numerose soluzioni out of the box, e quindi i tempi di consegna possono essere rispettati più facilmente.

Quindi i due requisiti per Node.js sono sostanzialmente **tempo** e **budget**. Inoltre vanno sempre valutate attentamente le richieste del cliente: un'applicazione che utilizza il machine learning per fornire un virtual shop assistant è completamente diversa da un semplice blog. Nel primo caso Node è un'ottima scelta, mentre nel secondo va bene anche un'installazione di WordPress.

In definitiva la scelta della tecnologia da usare non può mai prescindere dalle esigenze reali di un progetto web e non dovrebbe mai essere influenzata dalle tendenze correnti nello sviluppo web.