Node.js e i progetti web

In questo articolo vedremo di analizzare le funzionalità di Node.js in relazione alle varie tipologie dei progetti web.

Node.js è uno strumento tra i tanti a disposizione nello sviluppo web. Come tale, vi sono progetti le cui caratteristiche e specifiche si adattano meglio al design di Node, ed altri che richiedono invece soluzioni diverse.

Node.js non è stato concepito solo per il web come PHP. Il suo design lo rende agnostico rispetto all'uso di destinazione. Node supporta il protocollo HTTP, ma non offre delle API semplificate e dirette per creare un'applicazione web. ExpressJS, in tal senso, viene a colmare questa lacuna, fornendo supporto al routing HTTP e semplificando di fatto la gestione di un progetto.

In Node se non si ha l'ausilio di un modulo NPM specifico, ogni operazione deve essere implementata da zero, persino servire semplici file statici. Con Node si gestisce un server. A differenza di quanto avviene in PHP, dove le operazioni di base vengono gestite ad esempio da Apache o nginx, in Node si ha il controllo di tutte le operazioni, ma questo controllo totale implica il fatto che si dovrà scrivere del codice per la loro gestione. Se ad esempio si crea una directory con delle immagini in Apache, le immagini saranno immediatamente raggiungibili via HTTP. Viceversa in Node si dovrà scrivere del codice per poter servire quella directory e quei file tramite HTTP, operazione che ExpressJS semplifica tramite il metodo `express.static()`.

Quindi in Node è molto più complesso sviluppare *from scratch* rispetto ad altre soluzioni. Senza moduli NPM di supporto, lo sviluppo implica tempi maggiori a meno che chi sviluppa non disponga già di una sua libreria di strumenti pronti per l'uso.

Venendo alle tipologie di progetto, l'unico punto di riferimento certo è dato dallo **sviluppo di API REST**. Dato il supporto nativo di JavaScript al formato JSON e la moltitudine di moduli e tool disponibili, creare API REST in Node è qualcosa che di fatto viene offerto *out of the box*.

Per quanto riguarda invece progetti come blog, e-commerce, gestionali, CMS e CRM, dipende in gran parte dalla perizia tecnica di chi vuole affrontare queste implementazioni con Node.

In questo caso specifico, sono davvero poche le soluzioni già pronte disponibili. È anche da tenere presente che in questo specifico ambito la competizione con soluzioni realizzate in linguaggi come PHP, Python e Java risulta essere davvero problematica da gestire. Ad esempio se si vuole realizzare un blog, un potenziale cliente si aspetterebbe sicuramente di avere sia nel frontend che nel backend tutte le funzionalità già presenti in un CMS come WordPress.

Se si considera il fatto che la sola Media Library di WordPress con il suo sistema di upload e di gestione degli allegati unita all'inserimento di immagini, gallerie e immagine in evidenza da associare ai post ha alle sue spalle il lavoro di diversi sviluppatori del team di WordPress, appare chiaro che per uno sviluppatore full stack che operi da solo la mole di lavoro richiesta solo per questa feature è davvero notevole. Il discorso ovviamente cambia se ad affrontare la realizzazione del blog è un team di sviluppo perché in questo caso il carico di lavoro può essere ripartito tra più persone.

In conclusione, è necessario essere innanzitutto consapevoli del proprio livello di conoscenza di Node.js prima anche solo di pensare di affrontare un progetto per cui l'ecosistema di Node non offre soluzioni pronte.