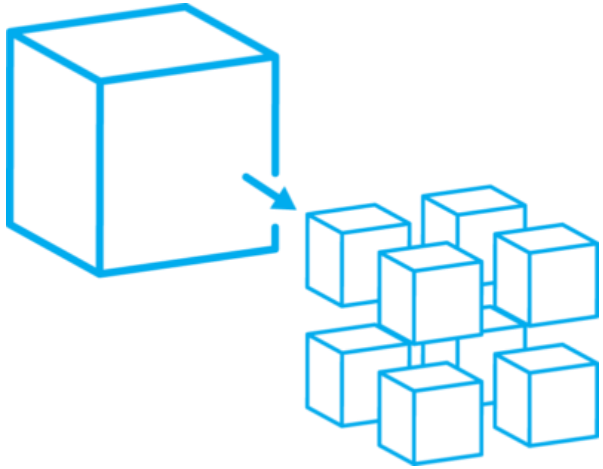
MICROSERVIZI

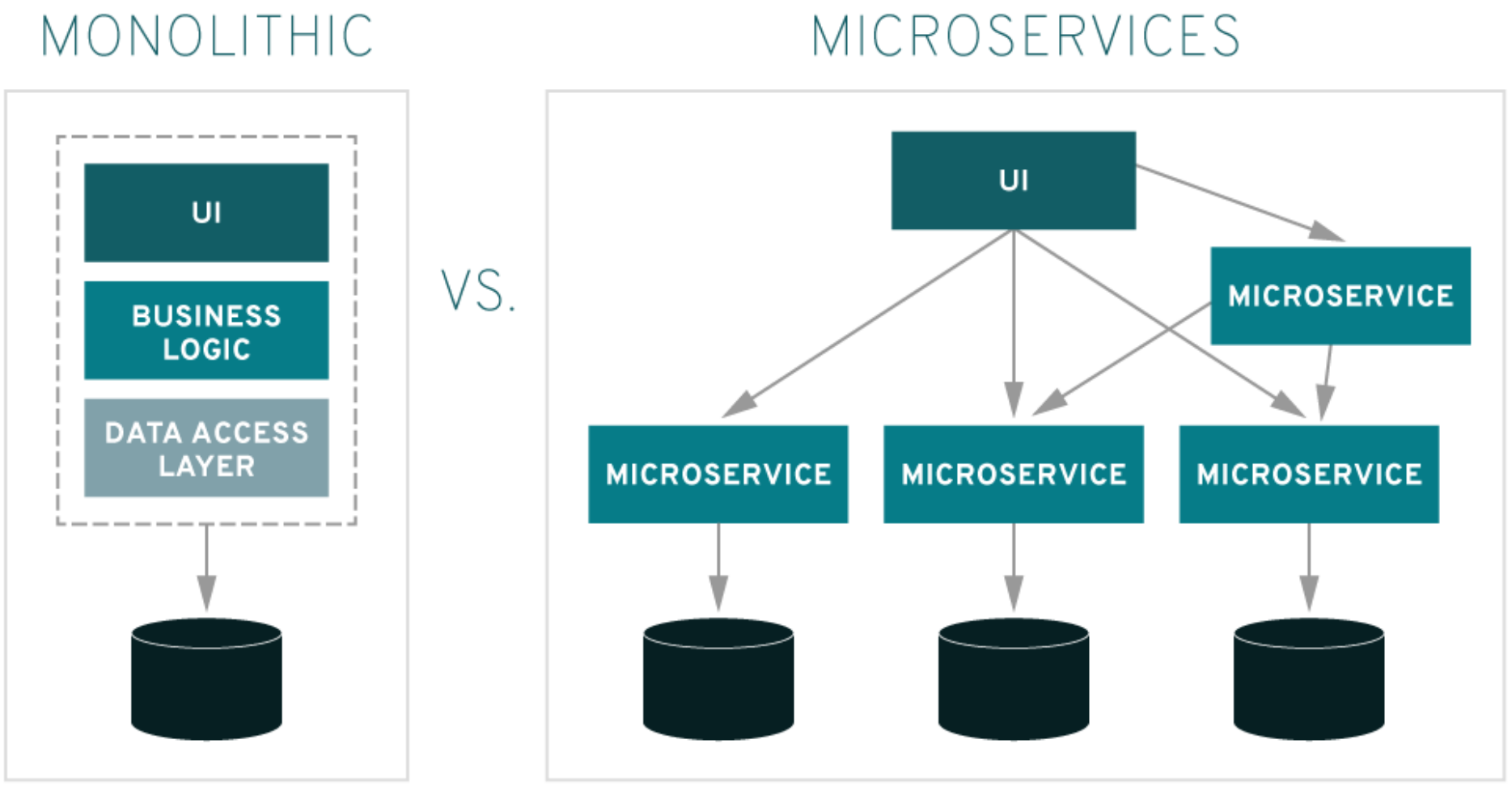
Riccardo Cattaneo



Introduzione

I **microservizi** sono un approccio per sviluppare e organizzare l'architettura dei software secondo cui quest'ultimi sono composti di servizi indipendenti di piccole dimensioni che comunicano tra loro tramite API ben definite. Questi servizi sono controllati da piccoli team autonomi.

Le architetture dei microservizi permettono di scalare e sviluppare le applicazioni in modo più rapido e semplice. E' un approccio ad una architettura IT che suddivide una singola complessa applicazione (APP Monolitica) in una serie di piccoli elementi indipendenti.



Microservizi

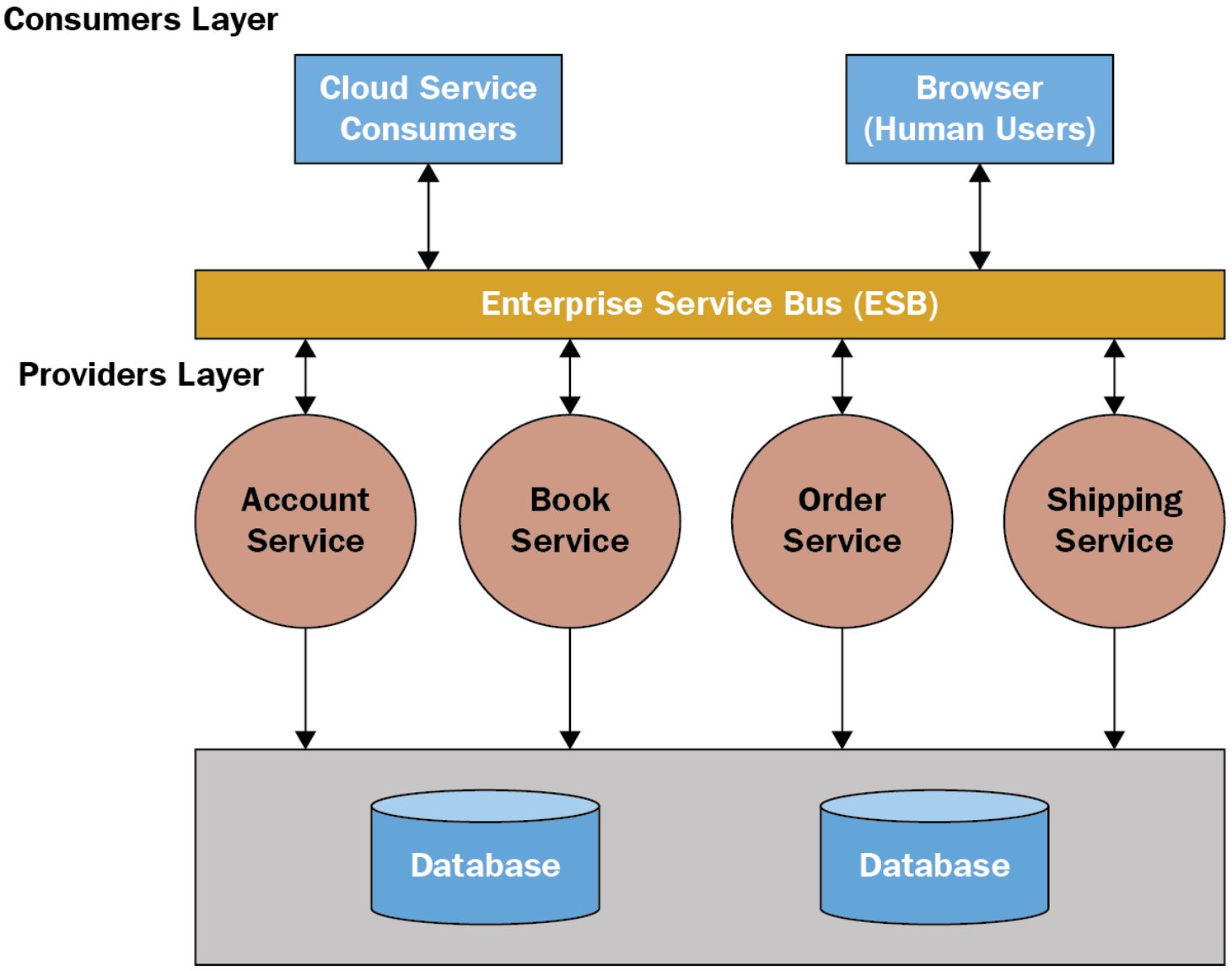
Nell’approccio monolitico, anche un cambiamento minimo a un software esistente imponeva un aggiornamento completo e un ciclo di controllo qualità a sé, che rischiava di rallentare il lavoro di tutti i team.

Tale approccio viene spesso definito “monolitico”, perché il codice sorgente dell’intera app veniva compilato in una singola unità di deployment (il classico war). Se gli aggiornamenti a una parte del software provocava errori, era necessario disconnettere tutto, fare un rollback totale del software. Questo approccio è ancora applicabile alle piccole applicazioni, ma le aziende in crescita non possono permettersi tempi di inattività.

Architettura SOA

Il primo tentativo di architettura a servizi nasce con l'architettura SOA, la quale possiamo definirla come la madre dell'architettura a microservizi.

Il concetto delle **S**ervice-**O**riented **A**rchitecture si afferma all'inizio degli anni Duemila come una collezione di servizi indipendenti che comunicano gli uni con gli altri tramite un Enterprise Service Bus (**ESB**). Prendiamo ad esempio il seguente esempio di architettura SOA :


Consumers Layer
Cloud Service Consumers
Browser (Human Users)
Enterprise Service Bus (ESB)
Providers Layer
Account Service
Book Service
Order Service
Shipping Service
Database
Database

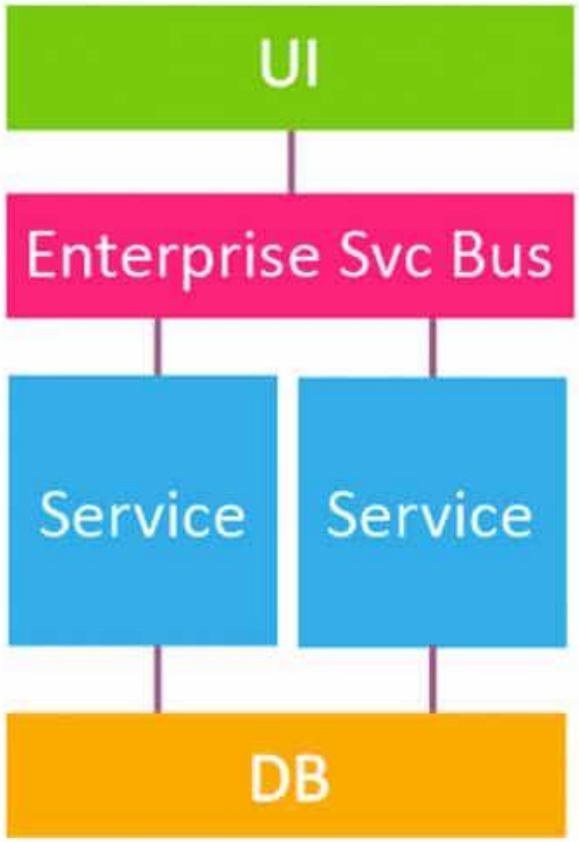
ESB

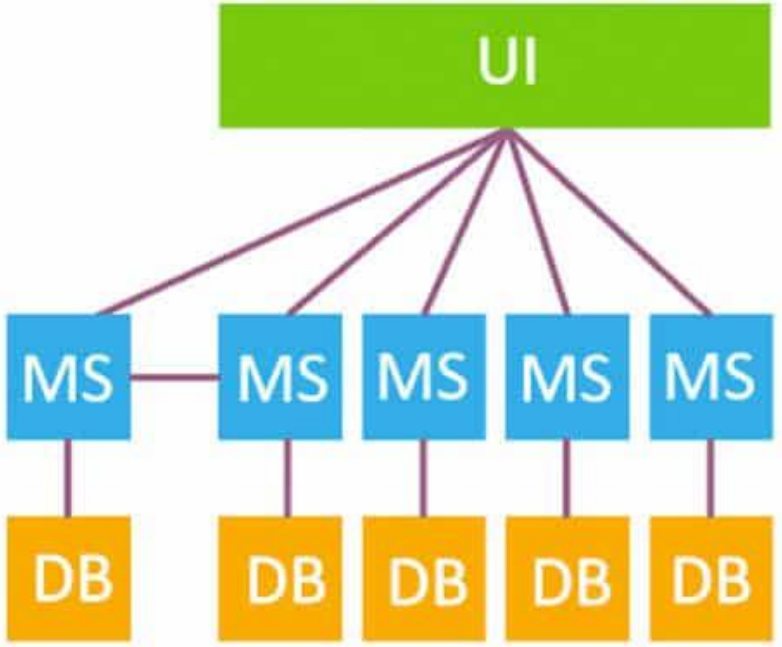
Un Enterprise Service Bus (ESB) è un'infrastruttura software che fornisce servizi di supporto a Service-Oriented Architecture complesse. Si basa su sistemi interconnessi con tecnologie eterogenee, e fornisce in maniera consistente servizi di coordinamento, sicurezza, messaggistica, instradamento intelligente e trasformazioni, agendo come una dorsale attraverso la quale viaggiano servizi software e componenti applicativi.

Un Enterprise Service Bus si contraddistingue come soluzione migliorativa, rispetto ad altre più classiche di tipo point-to-point in quanto ad esso sono delegati i servizi comuni denominati core service che andrebbero altresì realizzati.

Arrivati a questo punto possiamo chiederci : e quindi qual'è la differenza tra SOA e microservizi? La domanda è lecita, non a caso ho scritto che SOA può essere definita la madre dei microservizi; L'architettura a microservizi è una chiara evoluzione dell'architettura SOA.

Per evidenziare le differenze focalizziamoci sulla seguente immagine.


UI
Enterprise Svc Bus
Service
Service
DB


UI
MS
MS
MS
MS
MS
DB
DB
DB
DB
DB

Le differenze vengono immediatamente esaltate:

Granularità' dei servizi : in una tipica architettura SOA non si arriva neanche ad una decina di servizi mentre in un architettura a microservizi il numero dei servizi è molto più alto : Netflix ha dichiarato di avere più o meno 700 microservizi!

Dividendo la nostra architettura in molti servizi, ne consegue quindi che la dimensione del singolo servizio risulta essere molto ridotta rispetto ad un servizio SOA, da qui nasce appunto l'acronimo di microservizio.

Comunicazione : I microservizi abbandonano l'utilizzo di ESB, comunicando direttamente tra loro con meccanismi di comunicazione light. Nella SOA, l'ESB potrebbe diventare un singolo punto di errore che influisce sull'intero sistema. Poiché ogni servizio comunica attraverso l'ESB, se uno dei servizi rallenta, potrebbe ostruire l'ESB con le richieste per quel servizio. D'altra parte, i microservizi sono molto migliori nella tolleranza agli errori. Ad esempio, se un microservizio presenta un errore di memoria, verrà interessato solo quel microservizio. Tutti gli altri microservizi continueranno a gestire le richieste regolarmente.

Database : In SOA i servizi condividono gli storage mentre con i microservizi ogni servizio può avere un database indipendente.

Quando Cambiare

- **App orientate al Cloud** : l'applicazione usa tecnologie Cloud e deve dialogare con molti dispositivi;
- **Modifiche frequenti delle logiche** : I client richiedono sempre nuove caratteristiche e in tempi ridotti;
- **Crescente complessità e rigidità** : l'applicazione ha raggiunto livelli di complessità ingestibili;

Quando NON Cambiare

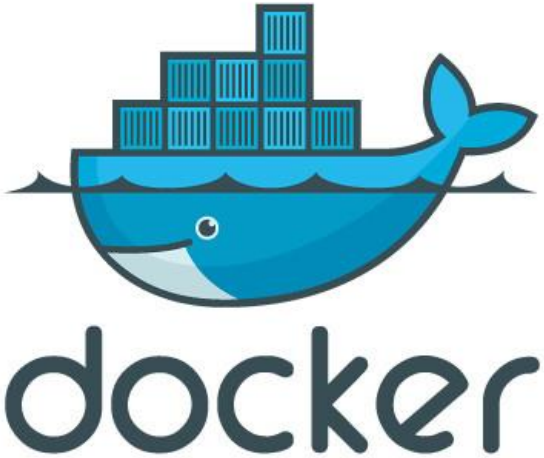
- **App indipendente dal Cloud** : l'applicazione usata in un contesto locale e non richiede l'uso del cloud;
- **Modifiche molto rare** : l'applicazione svolge le sue funzioni senza richiedere modifiche se non quelle di manutenzione.
- **Dimensioni ridotte** : un'applicazione di dimensioni ridotte, con funzionalità di base non richiede l'utilizzo dei microservizi.

Quale Tecnologia

I microservizi **non** sono ancorati ad una tecnologia, possono essere sviluppati con un qualsiasi linguaggio di programmazione come ad esempio C# , Phyton e Java.

Per un neofita si consiglia Java, in quanto è il linguaggio più utilizzato. Inoltre in Java esistono due framework specifici : **Spring Boot** e **Spring Cloud**, quest'ultimo include la flessibilità di Spring con l'aggiunta dello stack **Netflix**: un set di tool e librerie sviluppate da Netflix per aiutare ancor di più i programmatori a sviluppare software con un'architettura a microservizi.

NETFLIX



kubernetes

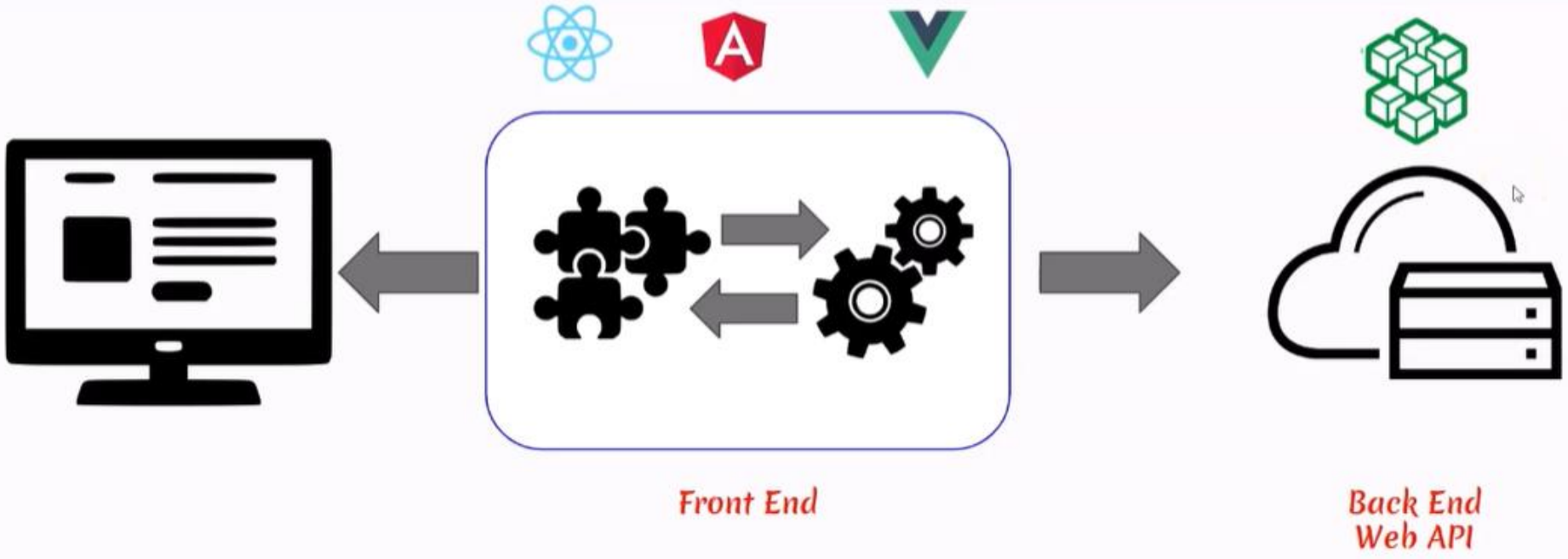
Altre 2 tecnologie che vedremo sono : Docker e Swarm.

- **Docker** : software che ci permette di creare dei contenitori dove troviamo tutte le risorse necessarie a far funzionare la nostra applicazione indipendentemente da dove la nostra applicazione viene installata (simile alle machine virtuali).
- **Swarm** : software che ci permette di "orchestrare" e gestire i nostri microservizi sia in locale che sul cloud

I Nostri Obiettivi

- Determinare quale tool usare per la creazione dei Microdervizi.
- Pianificare e Creare l'ecosistema di microservizi (WebAPI)
- Rendere i servizi Sicuri
- Centralizzare le chiamate alle Web API
- Garantire elevate livelli di prestazioni
- Dockenizzare i servizi (Docker)
- Eseguire il deploy nel Cloud (AWS)

Schema delle Moderne Web App



Ecosistema Microservizi