



MICROSERVIZI: INTRODUZIONE E CARATTERISTICHE

1 Comment / Di Dario Frongillo / In Design Pattern / 19 Ottobre 2019



Di Dario Frongillo
18 Ottobre 2019

1 Comment



MICROSERVIZI PIU' POPOLARI DELLA FERRAGNI

Si chiama microservice ed è il pattern architetturale più' popolare del momento.





Di Dario Frongillo

18 Ottobre 2019

1 Comment



Qualsiasi azienda si vanta di essere un'azienda che fa uso di microservice, qualsiasi sviluppatore scrive su linkedin tra le skill: microservice. La verità? Il grosso di loro bluffa: ad oggi conosco pochissime realtà italiane che utilizzano i microservice in produzione il resto confondono i microservice con un'architettura SOA. Il motivo? Nonostante questo stile architettonico abbia dei benefici indiscutibili, lavorare con i microservice è veramente difficile e costoso in termini di sviluppo e competenze. Per questo motivo ultimamente ho deciso di approfondire questo stile architettonico e con i miei appunti scrivere una serie di articoli. Partiremo con i principi fondamentali di un'architettura a

microservizi, approfondiremo nei prossimi articoli i pattern noti come SAGA con degli esempi pratici sviluppati con Spring Cloud.



Di Dario Frongillo

18 Ottobre 2019

1 Comment



Martin Fowler e James Lewis, due pilastri in materia danno la seguente definizione:

”



Di Dario Frongillo

18 Ottobre 2019

1 Comment



come una suite di piccoli servizi, ciascuno in esecuzione nel proprio processo e in comunicazione con meccanismi leggeri. Progettati e sviluppati intorno alle business capability del dominio, questi servizi possono essere deployati in modo indipendente e in modalità' totalmente automatizzata. La logica di questi servizi può' essere implementata con diversi linguaggi di programmazione e utilizzare diverse tecnologie di storage.

James Lewis e Martin Fowler

Senza dilungarci troppo su definizioni, andiamo dritti sulle caratteristiche che dovrebbe avere una buona architettura a microservizi.



Di Dario Frongillo

18 Ottobre 2019

1 Comment



Ai primordi dello sviluppo applicativo, anche un cambiamento minimo a un software esistente imponeva un aggiornamento completo e un ciclo di controllo qualità (QA, Quality Assurance) 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.

Sulla base di uno dei dogmi dell’ingegneria del software come il principio di **singola responsabilità** di Robert C. Martin,

”

Riunire le cose che cambiano per le



Di Dario Frongillo
18 Ottobre 2019

1 Comment

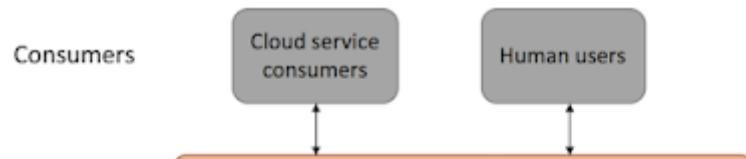


sono nati i primi studi di architetture a servizi.

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

ANALOGIE CON SOA

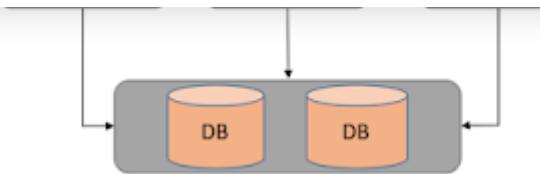
Il concetto delle Service-Oriented Architecture si afferma all'inizio degli anni Duemila come una collezione di servizi indipendenti che comunicano gli uni con gli altri tramite un Enterprise Service Bus (ESB). Prendiamo ad esempio il seguente esempio di architettura SOA





Di Dario Frongillo
18 Ottobre 2019

1 Comment



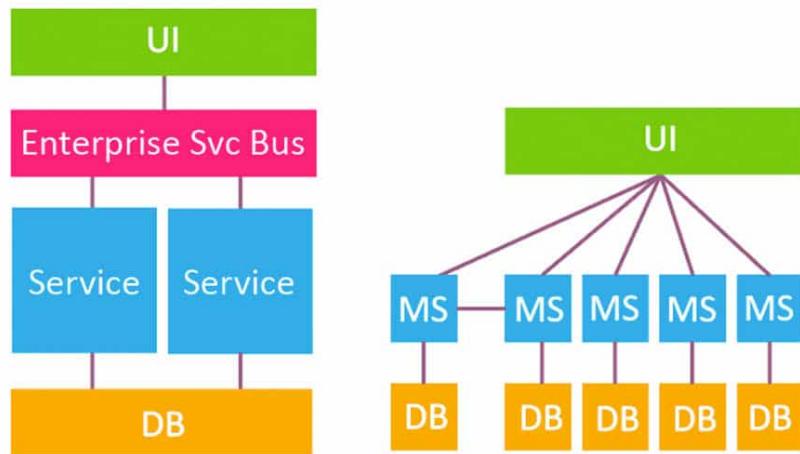
Possiamo constatare che esistono 2 ruoli principali in SOA, un fornitore di servizi e un consumatore di servizi. Un agente software può svolgere entrambi i ruoli. Il livello del consumatore è il punto in cui i consumatori (utenti umani, altri servizi o terze parti) interagiscono con la SOA e il livello del fornitore è costituito da tutti i servizi definiti all'interno della SOA.

I neofiti arrivati a questo punto dell'articolo potrebbero quindi chiedersi: e quindi qual'e' la differenza tra SOA e microservices ? La domanda e' lecita, non a caso ho scritto che SOA puo' essere definita la madre dei microservices; L'architettura a microservices e' una chiara evoluzione dell'architettura SOA sprinta dall'esigenza di una sempre più spinta scalabilità, la quale

permette di reggere il carico di milioni di utenti connessi in un determinato istante. Analizziamo quindi le differenze con SOA.



immagine.



Le differenze vengono immediatamente esaltate:

- **Granularità' dei servizi:** “Le dimensioni contano”. Sembra il motto di Rocco Siffredi ma non lo è; Martin Fowler in un suo famoso speech al GOTO, spiega che probabilmente il miglior modo di distinguere i due pattern è sulla base delle dimensioni; in una tipica architettura SOA non si arriva neanche ad una decina di servizi mentre in un architettura a microservice il numero dei servizi e' molto più alto: Netflix ha dichiarato di avere più o meno 700 microservices! Dividendo la nostra architettura in molti servizi, ne consegue quindi che le dimensioni addizionali diventano molto più grandi.



Di Dario Frongillo
19 Ottobre 2019

1 Comment



domanda. In letteratura si definisce un buon microservizio un servizio sufficientemente piccolo per adempiere correttamente ai principi di singola responsabilità, alta coesione, accoppiamento basso con altri servizi e un chiaro Bounded context (concetti approfonditi più avanti)

- **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 microservice ogni servizio può avere un database indipendente. La condivisione dei dati di un database tra servizi ha i suoi pro e contro. Ad esempio, i dati possono essere riutilizzati facilmente da tutti i servizi mentre porta dipendenza e accoppiamento stretto all'interno dei servizi e



Di Dario Frongillo
18 Ottobre 2019

1 Comment



esempio un architettura a microservizi di un magazzino:

- I dati dello stock sono salvati nel database del microservizio stock
- i dati dell'ordine sono invece salvati nel database del microservizio ordini.
- Un ordine riferisce ad un particolare stock.
- Per risolvere questo problema nel database degli ordini si terra' le informazioni degli ordini e l'identificativo univoco dello stock a cui punta un ordine.
- Il microservice degli ordini comunicherà con il microservice stock per avere l'intera rappresentazione di un dato stock (ad esempio tramite l'invocazione di un api rest del microservice stock).

Se vi interessa approfondire ulteriormente le differenze tra SOA e Microservice vi consiglio altamente di guardare il seguente video del mitico Martin Fowler

GOTO 2014 • Microservices • Martin Fowler



Di Dario Frongillo
18 Ottobre 2019

1 Comment



CARATTERISTICHE DI UNA ARCHITETTURA A MICROSERVIZI

Chiara l'analogia con SOA, passiamo quindi a introdurre le proprie' che caratterizzano una architettura a microservizi.

Abbiamo già' anticipato che un buon microservizio e' sufficientemente piccolo e deve coprire una singola responsabilità'.

Questo non basta..un buon microservizio deve anche rispettare i seguenti principi: accoppiamento lasco, coesione e Bounded Context, Analizziamo questi tre importanti concetti.

ACCOPIAMENTO LASCO (LOOSE COUPLING)



Di Dario Frongillo

18 Ottobre 2019

1 Comment



mostrarca a un servizio e distribuirlo, senza la necessita di modificare qualsiasi altro componente del sistema. Questo è davvero molto importante. Che cosa può causare un accoppiamento stretto? Un errore classico è scegliere uno stile di integrazione che lega strettamente un servizio ad un altro, facendo sì che i cambiamenti all'interno del servizio richiedano un cambiamento per i consumatori.

ALTA COESIONE

Se vogliamo fare una change vogliamo essere in grado di apportare modifiche in un solo servizio e rilasciare la nuova versione il prima possibile. Applicare una modifica su più servizi comporta il costo di dover rilasciare più servizi e il rischio di dimenticare di applicare la change su tutti i servizi coinvolti. La fase di progettazione ha quindi un ruolo fondamentale: occorre identificare i confini all'interno del nostro dominio in modo da dividere il dominio in N piccoli microservizi affinché' un determinato comportamento o responsabilità sia sviluppato

all'interno di un unico servizio. Per ottenere questo dobbiamo avere però chiaro il BOUNDED CONTEXT del nostro dominio.

BOUNDED CONTEXT



Di Dario Frongillo

18 Ottobre 2019

1 Comment



fornendo informazioni esplicite sui rispettivi limiti. Ogni contesto delimitato deve avere il proprio modello e il proprio database. Analogamente, ogni microservizio è proprietario dei rispettivi dati correlati. Ogni contesto delimitato inoltre ha un linguaggio comune specifico che semplifica le comunicazioni tra sviluppatori software ed esperti di dominio.

LA RISPOSTA AL GIUSTO CRITERIO DI COMPOSIZIONE DEI SERVIZI

Alta coesione, accoppiamento lasco e Bounded context sono quindi i tre criteri fondamentali presi in considerazione durante la progettazione di un architettura a microservizi. Una volta chiari questi concetti verrà' più' naturale suddividere il dominio applicativo in Bounded context sulla quale basare il dominio di un microservizio affinché ci sia alta coesione all'interno del servizio e disaccoppiamento lasco con gli altri microservizi.

ETEROGENEITÀ DELLE TECNOLOGIE



Di Dario Frongillo

18 Ottobre 2019

1 Comment



Queste sono le domande e considerazioni più stupide che leggo nelle community. Ragazzi i grandi sistemi come Netflix, Paypal e Facebook utilizzano architetture a microservizi con centinaia di microservizi sviluppati con una varietà di tecnologie e linguaggi che neanche immaginate. Questo perché essendo ogni microservice indipendente dall' altro si tende ad utilizzare la tecnologia giusta per quel componente. Quindi immaginatevi una grande piattaforma a microservizi:

- avete bisogno di un engine ad altissime prestazioni e realtime? Svilupperete il vostro microservizio ad es in C++ o Elixir
- Avete bisogno di un microservizio con algoritmi di machine learning? Utilizzate ad es Python per il vostro microservizio.

Stesso discorso per i database; si sceglie il tipo di database in base alla natura del singolo microservizio.

RESILIENZA

La gestione di errori imprevisti è uno dei problemi più difficili da



Di Dario Frongillo

19 Ottobre 2019

1 Comment



che eviti tempi di inattività o perdita di dati. Quindi come si può evitare che un errore di rete o di servizio si propaghi in cascata in altri servizi ? La soluzione : un client di servizio dovrebbe richiamare un servizio remoto tramite proxy che funziona in modo simile ad un interruttore automatico. Quando il numero di guasti consecutivi supera una soglia, l' interruttore di circuito scatta e per la durata di un periodo di timeout tutti i tentativi di richiamare il servizio remoto si interrompono immediatamente. Al termine del timeout, l' interruttore automatico consente il passaggio di un numero limitato di richieste di test. Se tali richieste hanno successo, l' interruttore ripristina il normale funzionamento. Altrimenti, se si verifica un errore, il periodo di timeout ricomincia. Questo pattern e' not come **circuit breaker** e ha il vantaggio che i servizi gestiscono il fallimento dei servizi che invocano, ma di contro è difficile scegliere i valori di timeout senza creare falsi positivi o introdurre eccessiva latenza

SCALABILITÀ'

Se la nostra architettura e' monolitica e abbiamo problemi di



Di Dario Frongillo

19 Ottobre 2019

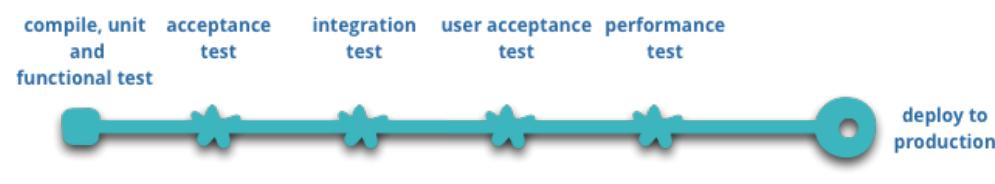
1 Comment



richiedono, permettendoci di eseguire altre parti del sistema su hardware con meno risorse e quindi meno potente. Ad esempio se nella nostra applicazione di magazzino sappiamo che il microservizio ordini è un componente critico e con molto traffico, possiamo scalare con più istanze di servizio il solo componente ordini che ha bisogno di più risorse hardware rispetto al microservizio magazzino.

DEPLOY FACILE, VELOCE E AUTOMIZZATO

Con i microservizi, possiamo apportare una modifica a un singolo servizio e distribuirlo indipendentemente dal resto del sistema con tecniche di continuos delivery del tutto automizzate.





Di Dario Frongillo

18 Ottobre 2019

1 Comment



velocemente e in maniera più sicura. Se si verifica un problema durante un rilascio, possiamo isolare rapidamente il servizio che da problemi, rendendo facile e veloce il rollback. Significa anche che possiamo rilasciare la nostra nuova funzionalità ai clienti più velocemente. Questo è uno dei motivi principali per cui organizzazioni come Amazon e Netflix utilizzano queste architetture, per garantire massima velocità di ripristino in caso di regressioni.

MIGLIORE ORGANIZZAZIONE DEI TEAM

Molti di noi hanno riscontrato i problemi associati a team di grandi dimensioni che lavorano su codebase altrettanto grandi. Questi problemi possono essere aggravati quando il team è distribuito in diversi settori e uffici. Sappiamo anche che i team più piccoli che lavorano su codebase più piccole tendono ad essere più produttivi.

I microservizi ci consentono di allineare meglio la nostra architettura alla nostra organizzazione, aiutandoci a ridurre al minimo il numero di persone che lavorano su qualsiasi codebase.



Di Dario Frongillo

18 Ottobre 2019

1 Comment



MICROSERVICE

Spero abbiate apprezzato questa introduzione ai microservice. Se vi siete incuriositi o avete bisogno per lavoro di approfondire questa architettura vi consiglio di leggere assolutamente:

- Building Microservices (Sam Newman)
- Microservices Pattern (Chris Richardson)
- <https://microservices.io/>
- [Martin flower blog](#)

CON CHE LINGUAGGIO INIZIARE

Sottolineo per l'ennesima volta che un architettura a microservizi puo' essere eterogenea dal punto di vista delle tecnologie e quindi potete utilizzare diversi linguaggi durante lo sviluppo dei vostri microservizi. Per un neofita vi consiglio Java, in quanto e' il linguaggio piu' utilizzato negli esempi che si trovano in letteratura. Inoltre in Java esiste un fantastico framework: Spring Cloud, il



Di Dario Frongillo
18 Ottobre 2019

1 Comment



Client Side Load Balancing (Ribbon). Pertanto nei prossimi articoli inizieremo a compiere i primi passi insieme con Spring Cloud.

MICROSERVICE LA PANACEA DI OGNI PROBLEMA ?

Il potenziale di questo stile architettonico è altissimo. L'architettura a microservizi si pone come soluzione architettonica ideale per:

- Software che hanno bisogno di un'alta scalabilità: Con l'esplosione del world wide web, una piattaforma di dominio pubblico raggiunge rapidamente bacini di migliaia se non milioni di utenti. I monoliti o architetture SOA faticano o scalano molto più difficilmente e con un costo molto più alto. Questo perché con un'architettura a microservizi riesci a scalare facilmente i microservizi più critici e quelli che hanno davvero bisogno di più risorse di calcolo senza dover scalare

l'intero software. I Cloud provider più noti offrono anche tecniche di auto-scaling utili per ottimizzare l'uso delle risorse di computing a seconda del traffico di quel momento

- Progetti con release frequenti e che devono essere veloci



Di Dario Frongillo

18 Ottobre 2019

1 Comment



- Ambiente Agile con piccoli team di sviluppo interfunzionali che sviluppano prodotti di grandi dimensioni in collaborazione

E' quindi la soluzione ideale per ogni software ? No, assolutamente no. Progettare e sviluppare un architettura a microservizi e' veramente complesso. Il livello di competenze richieste e' molto alto e mediamente poche aziende si possono permettere un team in grado di progettare e sviluppare un architettura a microservice:

- servono architect con alte competenze in grado di individuare i contesti e dividerli accuratamente in microservice con un giusto livello di granularità senza perdere di vista i principi che abbiamo riportato in questo articolo. Un errore in fase di progettazione e' letale!
- Nel team devono esserci figure altamente specializzate che non tutte le aziende si possono permettere: da software engineer che sanno sviluppare servizi scalabili con metodologie di CI&CD a figure altamente specializzate lato devOps, sistematico e cloud.



Di Dario Frongillo

18 Ottobre 2019

1 Comment



fase di build e deploy, in un architettura a microservice diventa un requisito obbligatorio se non vuoi far morire i tuoi sistemisti

- Poiché ogni servizio gira su processo separato, sono necessari strumenti di monitoraggio e manutenibilità per ciascun processo.
- La comunicazione inter process tra i microservizi del tuo software influisce negativamente sulle prestazioni del tuo software e sul carico della rete. Meccanismi di caching diventano fondamentali per migliorare le prestazioni.
I microservizi introducono problemi associati al calcolo distribuito come sicurezza, transazioni, concorrenza, ecc

Il mio modesto pensiero al riguardo ? Amo questa architettura ma se non lavori su piattaforme con un bacino di milioni di utenti e stai realizzando il classico gestionale o software enterprise con un bacino di qualche centinaio di utenti, progettare un architettura a microservice potrebbe essere “troppo” e potenzialmente potrebbe aumentare i costi dello sviluppo introducendo nuove difficoltà senza un chiaro beneficio. In questi casi un architettura a servizi “decisamente meno micro” potrebbe essere una soluzione

migliore. La morale e' quindi quella di valutare sempre il dominio e



Di Dario Frongillo

18 Ottobre 2019

1 Comment



microservice

pattern



A PROPOSITO DI ME



DARIO FRONGILLO

Fondatore di Italiancoders e dell'iniziativa devtalks.

Software architect con un forte background in Java,

Architettura REST, framework Spring , Database

Design, progettazione e sviluppo di SPA e RIA Web

application con framework Javascript. Attualmente

mi occupo di sviluppo soluzioni software in ambito

Banking e Finance: in particolare progettazione e

sviluppo di applicativi web based per la



Di Dario Frongillo

18 Ottobre 2019

1 Comment



Sponsored

Gli alleati naturali delle nostre difese immunitarie

Salute per Scharper



Di Dario Frongillo
18 Ottobre 2019

1 Comment



Le razze di cani più costose al mondo

Il Mondo dei Cani

Many failed before. Will you complete the Trial?

Hero Wars

Siediti prima di vedere la villa dove vive Maria De Filippi

RetroPages

Prima di incontrare persone, si raccomanda l'uso di questo prodotto antivirale scientificamente approvato

Agi

I reperti storici indicano agghiaccianti scoperte che riguardano l'umanità!

1 Commento **ItalianCoders** normativa sulla privacy Accedi ▾

1

Consiglia 1

Tweet

Condividi

Ordina dal più recente ▾



Partecipa alla discussione...

ENTRA CON

O REGISTRATI SU DISQUS

Nome



Miriano Esposito • un anno fa



Di Dario Frongillo

18 Ottobre 2019

1 Comment



successivamente passare a questo stile architettonurale. Mi raccomando nei prossimi articoli affronta bene le criticità dei MS, in particolare la gestione delle dipendenze e le tecniche di disaccoppiamento.

I NOSTRI PARTNER





Di Dario Frongillo

19 Ottobre 2019

1 Comment



MICROSERVIZI: INTRODUZIONE E CARATTERISTICHE

1 Comment / Di Dario Frongillo / In Design Pattern / 19 Ottobre 2019