

Programmazione Mobile

Nicola Noviello

nicola.noviello@unimol.it

Corso di Laurea in Informatica
Dipartimento di Bioscienze e Territorio
Università degli Studi del Molise
Anno 2023/2024

Basi di Architetture Orientate ai Servizi (SOA)

- Sistemi monolitici
- Introduzione al Service Oriented Computing
- Paradigma SOA e principi di progettazione dei servizi
- Micro-service architecture
- Introduzione ai servizi serverless



Introduzione alle architetture software

Sono la struttura fondamentale di un sistema informatico, definendo come le sue diverse componenti interagiscono e cooperano per raggiungere obiettivi comuni.

Evolgono continuamente per adattarsi alle crescenti esigenze di business e alle nuove tecnologie.



Architetture monolitiche

Architetture monolitiche

Struttura Monolitica

Un'unica applicazione che contiene tutti i componenti necessari per il suo funzionamento: frontend, backend, logica di business e database.

Implementazione Semplice

Semplice da implementare e deployare poiché non richiede la gestione di diversi servizi interconnessi.

Architetture monolitiche

Dipendenze Dirette

Tutte le funzionalità sono strettamente legate, rendendo difficile la modifica o la sostituzione di singoli componenti senza impatti sull'intero sistema.

Scalabilità Limitata

L'aumento del carico di lavoro richiede una scalabilità verticale dell'intero sistema, il che può diventare problematico e costoso nel tempo.

Vantaggi e svantaggi delle architetture monolitiche

- **Semplicità di implementazione e deployment:** Un'applicazione monolitica è più semplice da sviluppare e distribuire rispetto a un sistema più complesso.
- **Unità di sviluppo:** Lo sviluppo di un'applicazione monolitica coinvolge solitamente un team di sviluppatori che lavorano in sinergia sull'intero sistema.
- **Testabilità:** È più semplice testare un'applicazione monolitica nella sua interezza rispetto a un sistema distribuito.



Vantaggi e svantaggi delle architetture monolitiche

- **Sicurezza:** La concentrazione di tutta l'applicazione in un singolo componente può facilitare la gestione della sicurezza e dei permessi di accesso.
- **Scalabilità limitata:** La scalabilità di un'applicazione monolitica è spesso limitata, in quanto l'intero sistema deve essere scalato come un'unica unità.
- **Manutenibilità complessa:** Apportare modifiche o aggiornamenti a un'applicazione monolitica può essere più complicato a causa dell'accoppiamento stretto tra i componenti.



Architetture orientate ai servizi (SOA)

Service Oriented Computing (SOC)

Il SOC è un modello di sviluppo software basato sulla creazione di servizi autonomi e interoperabili, che possono essere combinati per soddisfare le esigenze aziendali. Questi servizi sono progettati per essere indipendenti dalle piattaforme e possono essere riutilizzati in diverse applicazioni.

I principali concetti sono i **servizi**, che rappresentano funzionalità atomiche, l'**interoperabilità**, che si riferisce alla capacità dei servizi di comunicare tra loro, e la **riusabilità**, la capacità di utilizzare i servizi in diverse situazioni.

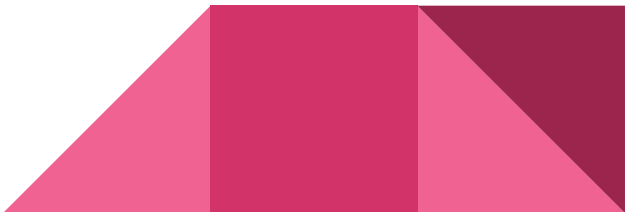
Un esempio può essere il servizio di pagamento online, che può essere riutilizzato in diverse applicazioni di e-commerce.



Architetture orientate ai servizi (SOA)

Le architetture orientate ai servizi (SOA) rappresentano un approccio di progettazione software in cui le funzionalità applicative sono esposte come servizi indipendenti e riutilizzabili. Questo paradigma consente una maggiore flessibilità, scalabilità e interoperabilità dei sistemi informatici.

In una SOA, le funzionalità sono incapsulate all'interno di servizi che comunicano tra loro attraverso interfacce ben definite, senza dipendere dalle implementazioni interne. Questo favorisce la modularità e la riusabilità del codice.



Principi e caratteristiche delle SOA

Modularità

L'architettura orientata ai servizi (SOA) si basa sulla creazione di componenti software indipendenti e modulari, detti servizi, che comunicano tra loro attraverso interfacce ben definite.

Riutilizzo

I servizi SOA possono essere riutilizzati in diverse applicazioni, aumentando l'efficienza e riducendo i costi di sviluppo.

Principi e caratteristiche delle SOA

Interoperabilità

Le applicazioni basate su SOA possono scambiare dati e funzionalità grazie a protocolli e standard comuni, come i web service.

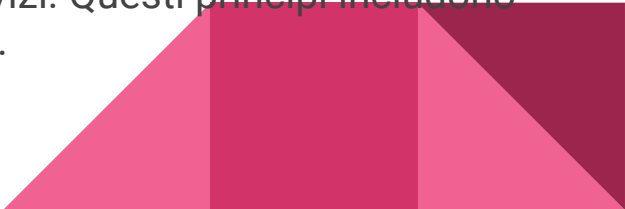
Flessibilità

L'approccio SOA consente di adattare rapidamente l'architettura alle esigenze aziendali in evoluzione, rendendo il sistema più agile e reattivo.

Componenti di un'architettura SOA

- **Servizi:** I servizi sono l'elemento fondamentale di un'architettura SOA e rappresentano funzionalità atomiche che possono essere richiamate tramite un'interfaccia ben definita. Ad esempio, un servizio di autenticazione può essere utilizzato per verificare le credenziali degli utenti
- **Registry:** I registri dei servizi sono repository centralizzati in cui vengono registrati e pubblicati i servizi disponibili. Questi registri consentono ai client di trovare e utilizzare i servizi necessari senza dover conoscere la loro implementazione specifica
- **Middleware di comunicazione:** Il middleware facilita lo scambio di messaggi tra i servizi all'interno di un'architettura SOA. Può includere tecnologie come i message broker, i protocolli di messaggistica e i servizi di orchestrazione
- **Gestione degli errori e sicurezza:** La gestione degli errori e la sicurezza sono aspetti critici di un'architettura SOA. È importante implementare meccanismi per gestire errori e timeout nei servizi, nonché per garantire l'autenticazione e l'autorizzazione degli utenti che accedono ai servizi

Caratteristiche di un'architettura SOA

- **Scalabilità dei servizi:** I servizi devono essere progettati in modo da poter scalare orizzontalmente per gestire carichi di lavoro variabili. Ciò può essere ottenuto utilizzando tecnologie come il load balancing, l'auto-scaling e la distribuzione del carico.
 - **Gestione delle prestazioni:** È importante monitorare le prestazioni dei servizi e identificare eventuali punti critici o bottleneck che potrebbero influenzare le prestazioni complessive del sistema. Gli strumenti di monitoraggio e di analisi dei dati possono aiutare a identificare e risolvere i problemi di prestazioni.
 - **Gestione della resilienza e tolleranza ai guasti:** È fondamentale progettare i servizi in modo da essere resilienti ai guasti e ai problemi di rete. Ciò può essere ottenuto utilizzando tecniche come il circuit breaking, il fallback e il retry dei servizi.
 - **Principi di sicurezza per i servizi:** La sicurezza è un aspetto critico di un'architettura SOA e deve essere considerata in tutte le fasi del ciclo di vita dei servizi. Questi principi includono la confidenzialità, l'integrità, l'autenticazione e l'autorizzazione.
- 

Architetture a microservizi

Architetture a microservizi

Servizi Indipendenti

I microservizi sono componenti software indipendenti e modulari, ciascuno con un proprio set di responsabilità ben definite.

Comunicazione Leggera

I microservizi interagiscono tra loro attraverso protocolli di comunicazione leggeri e interoperabili, come REST API o messaggistica asincrona.

Architetture a microservizi

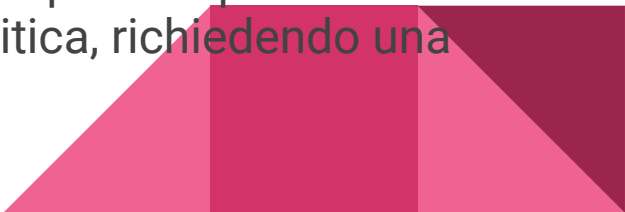
Scalabilità Flessibile

Ogni microservizio può essere scalato in modo indipendente in base alle proprie esigenze di carico, rendendo l'architettura altamente scalabile.

Indipendenza Tecnologica

I microservizi possono essere sviluppati e implementati utilizzando tecnologie diverse, consentendo maggiore flessibilità e indipendenza per i team di sviluppo.

Vantaggi e svantaggi dei microservizi

- **Modularità:** I microservizi offrono una struttura modulare che semplifica la gestione e lo sviluppo del software, permettendo di lavorare in modo indipendente su singole parti dell'applicazione
 - **Scalabilità:** Essendo indipendenti, i microservizi possono essere ridimensionati in modo flessibile in base alle esigenze, aumentando o diminuendo le risorse in modo mirato
 - **Resilienza:** Se un microservizio dovesse avere problemi, gli altri continuerebbero a funzionare senza essere influenzati, garantendo una maggiore affidabilità del sistema
 - **Complessità:** L'architettura a microservizi può risultare più complessa da progettare e gestire rispetto a un'applicazione monolitica, richiedendo una maggiore coordinazione tra i team.
- 

Architetture serverless

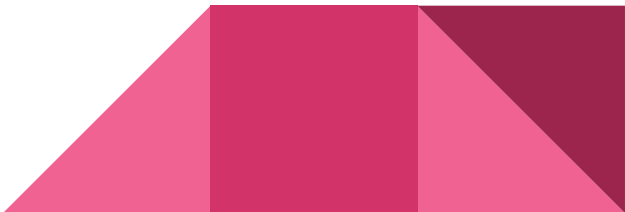
Architetture serverless

Le architetture serverless, note anche come Function as a Service (FaaS), eliminano la necessità di gestire direttamente l'infrastruttura sottostante. In questo modello, il fornitore di servizi cloud si occupa di scalare automaticamente le risorse in base al carico di lavoro, consentendo agli sviluppatori di concentrarsi esclusivamente sulla logica applicativa.

I vantaggi principali delle architetture serverless includono la riduzione dei costi di gestione dell'infrastruttura, la maggiore scalabilità e resilienza, oltre alla possibilità di implementare rapidamente nuove funzionalità senza preoccuparsi del provisioning dei server.



Confronto tra microservizi e serverless

- **Architettura:** I microservizi sono servizi autonomi e indipendenti, mentre il serverless si basa su funzioni senza server
 - **Scalabilità:** I microservizi offrono una scalabilità orizzontale più granulare, mentre il serverless si scala automaticamente in base alla domanda
 - **Complessità:** I microservizi comportano una maggiore complessità nella gestione e nel monitoraggio, mentre il serverless semplifica la complessità infrastrutturale
 - **Portabilità:** il serverless è spesso vincolato alle infrastrutture sulle quali è sviluppato
- 

Bibliografia e materiale consigliato

Bibliografia e materiale consigliato

- **Cosa si intende per SOA (architettura orientata ai servizi)?**
<https://aws.amazon.com/it/what-is/service-oriented-architecture/>
 - **Microservizi** <https://aws.amazon.com/it/microservices/>
 - **Qual è la differenza tra SOA e microservizi?**
<https://aws.amazon.com/it/compare/the-difference-between-soa-microservices/>
 - **Libro:** JOSUTTIS, Nicolai M. SOA in practice: the art of distributed system design. " O'Reilly Media, Inc.", 2007.
 - **Libro:** NEWMAN, Sam. Building microservices. " O'Reilly Media, Inc.", 2021.
- 