Architetture Software e Pattern

CODICE	**DT0320**
DURATA	**5 gg**
PREZZO	**1.990,00 €**
EXAM	

DESCRIZIONE

Il corso Architetture Software e Pattern offre una panoramica completa e approfondita dei principali pattern architetturali e delle metodologie di progettazione software utilizzate nello sviluppo moderno. L'obiettivo è fornire ai partecipanti le competenze necessarie per progettare sistemi software scalabili, resilienti, sicuri e facilmente manutenibili.

Progettato per sviluppatori, architetti software e Tech Lead, il percorso formativo accompagna dalla teoria all'implementazione pratica, con un approccio graduale che consente di comprendere e applicare i concetti fondamentali dell'Architettura Software moderna.

Durante , i partecipanti acquisiranno competenze avanzate nella progettazione di architetture, analizzando l'evoluzione dalle tradizionali Architetture Monolitiche ai più moderni approcci basati su Microservizi, Architetture Event-driven e soluzioni Cloud-native. Ogni modulo include esempi pratici, casi di studio e sessioni hands-on progettate per favorire l'applicazione immediata delle conoscenze acquisite nel proprio contesto professionale.

Grazie alla combinazione di teoria, pratica e best practice del settore, il corso rappresenta un riferimento per chi desidera migliorare le proprie capacità di progettazione e affrontare con successo le sfide dell'architettura software contemporanea.

Gli argomenti principali di questo corso intensivo saranno

- Panoramica delle Architetture Software
- Architettura Monolitica ed a Microservizi
- Comunicazione delle Architetture a Microservizi
- Gestione delle Transazioni
- Il Pattern SAGA
- Progettazione della Business Logic in una Architettura a Microservizi
- Implementazione di Query in una Architettura a Microservizi
- Pattern API Gateway

- Testing dei Microservizi
- Sviluppo di servizi per ambiente di produzione
- Deploy di Microservizi
- Refactoring dei Microservizi

Il corso combina teoria e pratica attraverso esercitazioni Hands-on, casi di studio reali e sessioni di laboratorio, permettendo ai partecipanti di applicare immediatamente le conoscenze acquisite

TARGET

- Sviluppatori Software che desiderano evolvere verso ruoli di progettazione e Architettura
- Architetti Software che vogliono aggiornarsi sulle moderne pratiche e Pattern
- Tech Lead e Team Leader responsabili di decisioni architetturali
- Professionisti IT

PREREQUISTI

- Solide conoscenze di programmazione in almeno un linguaggio moderno
- Familiarità con i concetti di base dello sviluppo software
- Esperienza di base con Database relazionali e NoSQL
- Conoscenze di base di Sistemi Distribuiti
- Comprensione dei principi di base del Cloud Computing

CONTENUTI

Introduzione alle architetture Software

- Cosa sono le Architetture Software
- Come si definisce l'architettura software di un sistema
- La fase della Raccolta dei requisiti
- La fase della Definizione degli obiettivi
- La fase di Analisi dei sistemi esistenti
- La fase di Selezione del pattern architetturale
- La fase di Identificazione dei componenti architetturali
- La fase di Definizione delle interfacce
- La fase del Disegno dei diagrammi architetturali
- La fase di Prototipazione e validazione
- La fase di Documentazione
- La fase di Iterazione e revisione
- La fase di Implementazione e monitoraggio
- La fase di Manutenzione e aggiornamento
- I principali Pattern Architetturali
- Overview sulla Architettura Monolitica
- Overview sulla Architettura a Strati

- Overview sulla Architettura a Microservizi
- Overview sulla Architettura Event-Driven
- Overview sulla Architettura a Servizi
- Overview sulla Architettura a Pipeline
- Overview sulla Architettura Client-Server
- Overview sulla Architettura Peer-to-Peer
- Overview sulla Architettura Model-View-Controller
- Overview sulla Architettura Model-View-ViewModel
- Overview sulla Architettura Broker
- Overview sulla Architettura Master-Slave
- Overview sulla Architettura Blackboard
- Overview sulla Architettura Microkernel
- Overview sulla Architettura Component-Based
- Overview sulla Architettura Domain-Driven Design
- Overview sulla Architettura Esagonale
- Overview sulla Architettura Command Query Responsibility Segregation
- Overview sulla Architettura Service Mesh
- Overview sulla Architettura Serverless
- Overview sulle Architetture moderne con il Cloud e DevOps

Dalla Architettura Monolitica a quella a Microservizi

- Cosa è l'Architettura Monolitica ?
- Design e Performance di una Architettura Monolitica
- Caratteristiche e vantaggi dell'Architettura Monolitica
- Limiti dell'Architettura Monolitica
- Esempi di utilizzo
- Come rendere agile una Architettura Monolitica
- L'Architettura dei Microservizi in soccorso a quella del Monolitico
- I Microservizi come forma di modularità
- Confronto tra Architettura a Microservizi e Service Oriented Architecture
- Cosa si intende per architettura orientata ai servizi ?
- Quali sono i vantaggi dell'architettura orientata ai servizi ?
- Principi base dell'Architettura orientata ai servizi - Interoperabilità
- Principi base dell'Architettura orientata ai servizi - Accoppiamento debole
- Principi base dell'Architettura orientata ai servizi - Astrazione
- Principi base dell'Architettura orientata ai servizi - Granularità
- Componenti dell'Architettura orientata ai servizi
- Come funziona l'Architettura orientata ai servizi
- Vantaggi e svantaggi dell'Architettura a Microservizi
- Il linguaggio dei Pattern dell'Architettura a Microservizi
- Pattern e linguaggi dei Pattern

- Panoramica del linguaggio dei Pattern dell'Architettura a Microservizi
- Organizzazione di sviluppo e distribuzione del Software
- Processo di sviluppo e distribuzione del Software
- Il lato umano dell'adozione dei Microservizi
- Definizione dell'Architettura a Microservizi di una Applicazione
- Cosa si intende per Decomposition strategies ?
- Identificare le operazioni di sistema
- Definire i servizi applicando il modello Decompose by Business Capability
- Definire i servizi applicando il modello Decompose by sub-domain
- Linee guida per la decomposizione
- Ostacoli alla scomposizione di una Applicazione in Servizi
- Definizione delle API di Servizio

Comunicazione delle Architetture a Microservizi

- Panoramica della comunicazione "interprocesso" in una Architettura a Microservizi
- Stili di interazione
- Definizione delle API in una Architettura a Microservizi
- Evoluzione delle API
- Formati dei messaggi
- Comunicazione tramite il modello di invocazione di procedure remote sincrone
- Utilizzo di REST
- Utilizzo di gRPC
- Gestione di Failure parziali tramite il Pattern Circuit Breaker
- Comunicazione tramite il Pattern Asynchronous Messaging
- Panoramica della messaggistica
- Implementazione degli stili di interazione tramite messaggistica
- Creazione di una specifica API per un'API di servizio basata sulla messaggistica
- Utilizzo di un Broker di messaggi
- Gestione dei messaggi duplicati
- La Messaggistica Transazionale
- Librerie e Framework per la messaggistica
- Utilizzo della messaggistica asincrona per migliorare la disponibilità
- La comunicazione sincrona riduce la disponibilità
- Eliminazione dell'interazione sincrona

Gestire le Transazioni con il Pattern SAGA

- Definizione e origini del Pattern SAGA
- Problematiche risolte dal pattern SAGA
- Principi di progettazione ACID vs BASE
- Consistenza eventuale vs consistenza forte

- Confronto con il Pattern Two-Phase Commit (2PC)
- Anatomia di una SAGA, Transazioni locali e azioni di compensazione
- Approcci Implementativi del Pattern SAGA
- Orchestrazione vs Coreografia
- Caratteristiche, vantaggi e svantaggi della Orchestrazione
- Caratteristiche, vantaggi e svantaggi della Coreografia
- Criteri di scelta dell'approccio più adatto
- SAGA basate su eventi
- Architetture event-driven
- Sistemi di Message Broker come Kafka e RabbitMQ
- Gestione degli eventi e Pattern di pubblicazione/sottoscrizione
- SAGA Orchestrate
- Servizio Orchestratore, responsabilità e implementazione
- Workflow e state machine
- Gestione del contesto di esecuzione
- Gestione degli errori con SAGA
- Consistenza dei dati ed Overview dela Pattern CQRS
- Osservabilità e tracciamento distribuito con SAGA
- Framework e librerie per il Pattern SAGA
- Spring Cloud Stream/Netflix Conductor
- Axon Framework
- Apache Camel
- Integrazione con Infrastrutture Cloud AWS, Azure e Google
- Implementazioni con Kubernetes
- Gestione delle Transazioni in una Architettura a Microservizi
- La necessità di Transazioni distribuite in una Architettura a Microservizi
- Il problema delle Transazioni distribuite
- Utilizzo del Pattern Saga per mantenere la coerenza dei dati
- Gestione della mancanza di isolamento
- Contromisure per la gestione della mancanza di isolamento

Il Domain Driven Design

- Cos'è Domain Driven Design
- DDD è una metodologia ?
- DDD è un insieme di pattern ?
- DDD è una tecnica di gestione della complessità ?
- DDD è una strategia di gestione progetti ?
- DDD è un insieme di strumenti utili ?
- Definiamo cosa è un DDD
- Il Domain Model Pattern
- Anemic Domain Model

- Ereditarietà e classi astratte
- Perchè un modello ?
- Lo scopo della nostra applicazione
- Londra: Rappresentazione 1
- Londra: Rappresentazione 2
- Londra: Rappresentazione 3
- Il Dominio dell'applicazione
- I Confini del Dominio
- L'Ecosistema per il DDD
- La comunicazione con l'esperto di dominio
- Ubiquitous language
- Quale formalismo usare ?
- Il codice è il modello
- La nostra comprensione del Dominio
- Esplorare le soluzioni

Progettazione della Business Logic in una Architettura a Microservizi

- Modelli di organizzazione della Business Logic
- Progettazione della Business Logic col Pattern Transaction script
- Progettazione della Business Logic col Pattern Domain Model
- Informazioni sulla progettazione basata sul Dominio
- Progettazione di un modello di dominio utilizzando il modello di aggregazione DDD
- Il problema dei confini Fuzzy
- Regole di aggregazione
- Granularità degli aggregati
- Progettazione della Business Logic con aggregati
- Pubblicazione di eventi di Dominio
- Cos'è un evento di Dominio ?
- Identificare gli eventi di Dominio
- Generazione e pubblicazione di eventi di Dominio
- Sviluppo della Business Logic tramite Event Sourcing
- Il problema della persistenza tradizionale
- Panoramica dell'Event Sourcing
- Gestione degli aggiornamenti simultanei tramite l'Optimistic Locking
- Event Sourcing e pubblicazione di eventi
- Utilizzo di Snapshot per migliorare le Performance
- Vantaggi e svantaggi dell'Event Sourcing
- Introduzione al Framework Client Eventuate per Java e Spring
- Utilizzo di SAGA ed Event Sourcing insieme

- Implementazione di SAGA basata sulla Coreografia tramite Event Sourcing
- Creazione di SAGA basata sull'Orchestrazione

Implementazione di Query in una Architettura a Microservizi

- Query utilizzando l'API Composition Pattern
- Cosa si intende per API Composition Pattern
- L'operazione findOrder()
- Implementare findOrder() con l'API Composition Pattern
- Problemi di progettazione dell'API Composition Pattern
- Vantaggi e svantaggi dell'API Composition Pattern
- Utilizzo del Pattern CQRS e perchè utilizzarlo
- Panoramica, vantaggi e svantaggi del Pattern CQRS
- Progettazione di viste CQRS
- Scelta di un Datastore
- Progettazione del modulo di accesso ai dati
- Aggiunta e aggiornamento di viste CQRS
- Implementazione di una Vista CQRS con AWS DynamoDB
- Modulo OrderHistoryEventHandlers
- Modellazione dei dati e progettazione delle Query con DynamoDB
- La Classe OrderHistoryDaoDynamoDb

Lo Schema Architetturale CQRS

- Introduzione al CQRS
- Le sfide di mantenere un unico modello per due contesti
- Una Architettura migliore per contesti delimitati complessi
- L'Explicitly Modeling Intent
- Un modello libero dalle distrazioni della presentazione
- Handling di una Business Request
- Query Side ed il Domain Reporting
- Report mappati direttamente al modello di dati
- Viste materializzate costruite da eventi di dominio
- I pregiudizi sul CQRS
- CQRS è complesso ?
- CQRS è alla fine consistente ?
- CQRS è alla fine coerente ?
- I modelli devono avere come base gli eventi ?
- I commands dovrebbero essere asincroni ?
- CQRS funziona solo con i sistemi di messaggistica ?
- È necessario utilizzare gli eventi di dominio con CQRS ?

Pattern API Gateway

- Cosa è un Pattern API Gateway
- La sua importanza
- Bassa Latenza
- Gestione del traffico
- Le Policy di limitazione della frequenza
- Le Policy di limitazione delle richieste
- Le Policy di controllo delle operazioni simultanee
- Le Policy sulle interruzioni
- Il bilanciamento dinamico del carico
- Problemi di progettazione API Gateway
- Vantaggi e svantaggi di un Pattern API Gateway
- Netflix come esempio di API Gateway
- Implementazione di un API Gateway
- Utilizzo di un prodotto/servizio API Gateway API Standard
- Sviluppo di un API Gateway personalizzato
- Implementazione di un API Gateway tramite GraphQL
- Interazione delle API Gateway con Kubernetes
- In che modo un'API Gateway API supporta ambienti DevOps e Serverless

Testing dei Microservizi

- Strategie di Test per architetture di microservizi
- Panoramica dei Test
- La sfida del Test dei Microservizi
- La Pipeline di distribuzione
- Scrittura di test unitari per un servizio
- Sviluppo di test unitari per entità
- Scrittura di test unitari per oggetti valore
- Sviluppo di test unitari per SAGA
- Scrittura di test unitari per servizi di dominio
- Sviluppo di test unitari per controller
- Scrittura di test unitari per gestori di eventi e messaggi
- Scrittura di test di integrazione
- Test di integrazione di persistenza
- Test di integrazione di interazioni in stile richiesta/risposta basate su REST
- Test di integrazione di interazioni in stile pubblicazione/sottoscrizione
- Test di integrazione per interazioni asincrone richiesta/risposta
- Sviluppo di test dei componenti
- Definizione di test di accettazione
- Scrittura di test di accettazione tramite Gherkin

- Progettazione di test dei componenti
- Scrittura di test dei componenti per il servizio ordini FTGO
- Scrittura di test end-to-end
- Progettazione di test end-to-end
- Scrittura di test end-to-end
- Esecuzione di test end-to-end

Sviluppo di servizi per ambiente di produzione

- Sviluppo di servizi sicuri
- Panoramica della sicurezza in una Applicazione monolitica tradizionale
- Implementazione della sicurezza in una Architettura a Microservizi
- Progettazione di servizi configurabili
- Utilizzo della configurazione esternalizzata basata su Push
- Utilizzo della configurazione esternalizzata basata su Pull
- Progettazione di servizi osservabili
- Utilizzo del modello API Health Check
- Applicazione del modello Log Aggregation
- Utilizzo del modello Distributed Tracing
- Applicazione del modello Application Metrics
- Utilizzo del modello Exception Tracking
- Applicazione del modello Audit Logging
- Modello di progettazione Chassis
- Sviluppo di servizi tramite il modello Microservice Chassis
- Utilizzo di uno Chassis di Microservizi
- Dal Chassis di microservizi al Service Mesh

Deploy di Microservizi

- Deploy di servizi tramite il Pattern Language-specific packaging format
- Vantaggi e svantaggi del servizio come Pattern Language-specific packaging format
- Deploy di servizi tramite il Pattern VM
- Vantaggi e svantaggi del Deploy di Pattern VM
- Distribuzione di servizi tramite i Container
- Distribuzione di servizi tramite Docker
- Vantaggi e svantaggi del Deploy di Container
- Distribuzione di API Gateway
- Utilizzo di un Service Mesh per separare il Deploy dal rilascio
- Deploy di servizi tramite il Pattern Serverless
- Panoramica del Deploy Serverless con AWS Lambda
- Sviluppo di una funzione Lambda
- Invocazione di funzioni Lambda

- Vantaggi dell'utilizzo di funzioni Lambda
- Svantaggi dell'utilizzo di funzioni Lambda
- Distribuzione di un servizio RESTful tramite AWS Lambda e AWS Gateway

Refactoring dei Microservizi

- Panoramica del Refactoring in Microservizi
- Perché effettuare il Refactoring di una Architettura Monolite ?
- Strategie per il Refactoring da Monolite in Microservizi
- Implementazione di nuove funzionalità come servizi
- Separazione del Presentation Layer dal Backend
- Estrazione delle capacità aziendali nei servizi
- Progettazione del modo in cui il servizio e il monolito collaborano
- Mantenimento della coerenza dei dati tra un servizio e un Monolite
- Gestione dell'autenticazione e dell'autorizzazione
- Progettazione del Delayed Delivery Service
- Scomposizione del Monolite
- Panoramica del Delivery Service
- Progettazione del modello di dominio del Delivery Service