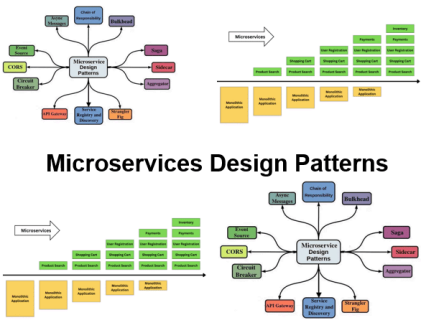
Microservices Design Patterns

Duration: 4 days **Price: € 2999**

Open Schedule



Audience Microservices Design Patterns

The course Microservices Design Patterns is intended for senior developers and software architects who want to implement design patterns in a microservices architecture.

Prerequisites Course Microservices Design Patterns

Good understanding of software development concepts and distributed systems. Experience with cloud platforms and containers is beneficial for understanding.

Realization Training Microservices Design Patterns

Demos under guidance of the trainer, alternated with presentations, discussions of case studies and practical exercises.

Certificate Microservices Design Patterns

After successfully completing the course, attendants will receive a certificate of participation Microservices Design Patterns.

Content Course Microservices Design Patterns

In the course Microservices Design Patterns, participants learn which design patterns can be applied in a Microservices Architecture. The structure and applicability of the different design patterns is discussed in depth.

Intro Microservices

This module introduces microservices as an architectural style. It compares monolithic and microservices architectures, highlighting benefits like scalability, cohesion, and independence, while addressing challenges like complexity and interdependence. Key principles include single responsibility, minimal coupling, and high cohesion.

Architecture Patterns

Participants explore common system design and microservices architecture patterns. Topics include layered architectures, separation of concerns, RESTful communication, Backend for Frontend, and Micro Frontends.

API Gateway Pattern

This module explains the API Gateway pattern as a facade and reverse proxy. It serves as a single entry point for clients, supports request aggregation and routing, and integrates with service registries for dynamic discovery.

Database per Service

This module focuses on each microservice managing its own database. Key topics include polyglot persistence, independent scalability, data encapsulation, and the avoidance of the shared database anti-pattern.

Saga Pattern

Here, participants learn how to handle distributed transactions using the Saga pattern. It includes concepts like compensating transactions, choreography vs orchestration, and maintaining consistency without two-phase commit.

Aggregator Pattern

The aggregator pattern allows combining responses from multiple microservices. The module covers variations like scatter-gather, chaining, branching, comparison proxies, and the importance of service discovery.

Circuit Breaker Pattern

This module teaches how the circuit breaker pattern prevents cascading failures in microservices. It explains availability handling, failure isolation, and the various circuit states: open, closed, and half-open.

Command Query Segregation

Participants learn about the CQRS pattern, where read and write operations are separated. It avoids inefficient joins and allows different storage models for queries and commands.

Asynchronous Messaging

This module discusses inter-service communication using messaging. Topics include publish-subscribe models, message brokers like RabbitMQ and Kafka, the AMQP protocol, and how async messaging supports scalability and decoupling.

Modules Course Microservices Design Patterns

Module 1: Intro Microservices	**Module 2: Architecture Patterns**	**Module 3: API Gateway Pattern**
What are Microservices? Monolith versus Microservices Benefits of Microservices Challenges of Microservices Single Responsibility Minimize Coupling Maximize Cohesion Scalability	System Design Patterns Layered Architectures Separation of Concern Microservices Patterns Synchronous Communication Using REST and HTTP Backend for Frontend Micro Frontends	What is an API Gateway? Facade Functionality Reverse Proxy Single Entry Point Requests Aggregation Request Routing Service Registry Service Discovery
Module 4: Database per Service	**Module 5: Saga Pattern**	**Module 6: Aggregator Pattern**
Dedicated Databases Separation of Concerns Independent Data Management Polyglot Persistence Independent Scaling Data Encapsulation Reducing Coupling Shared Database Anti-Pattern	Transaction Handling Distributed Transactions Two Phase Commit Maintaining Data Consistency Compensating Transactions Saga Coordination Saga Choreography Saga Orchestration	Distributing Requests Aggregating Results Scatter Gather Variation Chained Variation Multiple Chains Branch Variation Comparison Proxy Pattern Using Service Discovery
Module 7: Circuit Breaker Pattern	**Module 8: Command Query Segregation**	**Module 9: Asynchronous Messaging**
Need for Circuit Breaking Failing Microservices High Availability Preventing Downtime Circuit Barrier Preventing Cascade Failure Circuit Breaker States Open, Closed, Half Open	CQRS Pattern Separate Operations Avoid Complex Queries Prevent Inefficient Joins Read versus Update Commands for Update Queries for Read Different Databases	Interprocess Communication Asynchronous Communication Backend Internal Microservices DIP Principle Publish and Subscribe Using Message Brokers async AMQP protocol Rabbit MQ and Kafka
Module 10: Event Sourcing	**Module 11: Strangler Pattern**	**Module 12: Decomposition Patterns**
Storing Events Single Source of Truth Sequential Event List Materialized Views Denormalized Views Splitting Databases Replaying Events Increase Query Performance	Legacy System Modernization Application Migration Evolve Gradually Avoid Bing Bang Rewrites Resource Utilization Risk Management Implementing Strangulation API Gateway as Proxy	Decomposing Microservices By Business Capability By Subdomain Domain Driven Design Bounded Context Pattern Propagating Cohesiveness Strategic DDD Tactical DDD