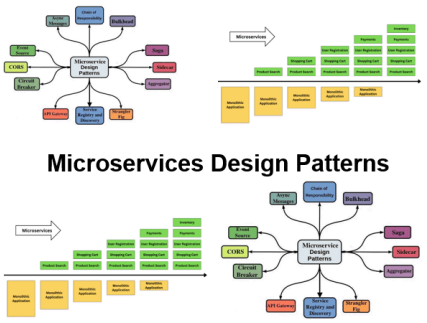
Microservices Design Patterns

Duur: 4 dagen **Prijs: € 2999**

Open Rooster



Doelgroep Cursus Microservices Design Patterns

De cursus Microservices Design Patterns is bestemd voor senior developers en software architecten die design patterns in een microservices architectuur willen toepassen.

Voorkennis Cursus Microservices Design Patterns

Goed begrip van software development concepten en gedistribueerde systemen. Ervaring met cloud platforms en containers is bevorderlijk voor de begripsvorming.

Uitvoering Training Microservices Design Patterns

Demos onder leiding van de trainer afgewisseld met presentaties, besprekingen van case studies en praktische oefeningen.

Certificaat Microservices Design Patterns

De deelnemers krijgen na het goed doorlopen van de cursus een certificaat van deelname aan Microservices Design Patterns.

Inhoud Cursus Microservices Design Patterns

In de cursus Microservices Design Patterns leren de deelnemers welke design patterns kunnen worden toegepast in een Microservices Architectuur. De structuur en toepasselijkheid van de verschillende design patterns wordt diepgaand behandeld.

Intro Microservices

Deze module introduceert microservices als architecturaal paradigma. Er wordt een vergelijking gemaakt met monolithische architecturen, en voordelen zoals schaalbaarheid, cohesie, en onafhankelijkheid worden besproken. Tegelijkertijd worden de uitdagingen zoals complexiteit en afhankelijkheden belicht. Belangrijke concepten zijn single responsibility, minimale koppeling, en maximale cohesie.

Architectuurpatronen

In deze module worden architectuurpatronen voor microservices behandeld. Dit omvat gelaagde architecturen, het scheiden van verantwoordelijkheden, en communicatiepatronen zoals REST en HTTP. Specifieke patronen zoals Backend for Frontend en Micro Frontends worden ook besproken.

API Gateway Patroon

De module behandelt het gebruik van een API Gateway als façade en reverse proxy. Het dient als een enkel toegangspunt voor externe clients en ondersteunt functies zoals aggregatie van verzoeken, routering naar interne services, en het gebruik van service registry en discovery.

Database per Service

Hier leren deelnemers hoe elke microservice zijn eigen database kan beheren. Onderwerpen zoals polyglot persistence, onafhankelijke schaalbaarheid, data-encapsulatie en het vermijden van het shared database anti-pattern staan centraal.

Saga Patroon

Deze module gaat over het beheren van gedistribueerde transacties met behulp van het Saga-patroon. Thema's zijn onder andere compensatietransacties, choreografie en orkestratie, en consistentie tussen services zonder het gebruik van two-phase commit.

Aggregator Patroon

Het aggregator patroon helpt bij het combineren van gegevens uit meerdere microservices. Varianten zoals scatter-gather, chained aggregatie, branch variaties en het comparison proxy patroon worden besproken. Service discovery speelt hierin ook een rol.

Circuit Breaker Patroon

Deelnemers leren hier hoe het circuit breaker patroon helpt om uitval van microservices te voorkomen. Onderwerpen zijn beschikbaarheid, cascaderende uitval vermijden, en de verschillende toestanden van een circuit breaker: open, gesloten, half-open.

Command Query Segregation

Deze module introduceert het CQRS-patroon waarbij lees- en schrijfbewerkingen worden gescheiden. Dit voorkomt inefficiënte joins en maakt gebruik van verschillende opslagmethodes voor lees- en schrijfoperaties. Command- en queryverantwoordelijkheden worden

Event Sourcing

In deze module leren deelnemers hoe event sourcing werkt als alternatief voor traditionele database-opslag. Gebeurtenissen worden

Decompositiepatronen

De laatste module bespreekt hoe microservices opgesplitst kunnen worden op basis van business capabilities, subdomeinen en bounded contexts. Domain Driven Design, zowel strategisch als tactisch, vormt de kern van deze decompositieaanpak.

Modules Cursus Microservices Design Patterns

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

SpiralTrain BV
Standerdmolen 10, 2e verdieping
3995 AA Houten

info@spiraltrain.nl
www.spiraltrain.nl
Tel.: +31 (0) 30 – 737 0661

Locaties
Houten, Amsterdam, Rotterdam, Eindhoven, Zwolle, Online