MICROSERVICES MİMARİSİ EĞİTİMİ

2 GÜN

Digital Vizyon
Akademi

www.digitalvizyon.net

İçindekiler

Eğitim Hakkında

"Microservices Mimarisi Eğitimi", microservices mimarisi üzerinde kapsamlı bir eğitim sunar. Bu eğitim, microservices tabanlı uygulamaların tasarımı ve geliştirilmesinde gereken temel konuları ele alır.

Eğitim, microservices mimarisi prensiplerini, veri yönetimi, hata toleransı, mesajlaşma ve diğer kilit kavramları öğretir. Katılımcılar, gerçek hayattan örnekler ve projeler aracılığıyla microservices mimarisi konusundaki bilgilerini artırırlar.

Eğitim ayrıca, uygulamaların microservices tabanlı olarak tasarlanması ve geliştirilmesi sürecinde hangi araç ve teknolojilerin nasıl kullanılabileceğini de öğretir. Katılımcılar, büyük veri işleme, real-time analiz, hata toleransı ve diğer özelliklerin nasıl yönetileceğini öğrenirler. Ayrıca, microservices mimarisi tabanlı uygulamaların nasıl tasarlanıp yönetileceğini de anlarlar.

"Microservices Mimarisi Eğitimi", microservices tabanlı uygulama tasarımı ve geliştirme sürecinin tümünü kapsar. Katılımcılar, modern uygulamalar tasarlamaya ve geliştirmeye başlamadan önce ihtiyaç duyacakları temel becerileri kazanırlar. Eğitim, örnekler, pratik uygulamalar ve projeler yoluyla öğrenmeyi sağlar ve katılımcıların microservices tabanlı uygulama tasarlamaya ve geliştirmeye başlamalarına yardımcı olur.

Eğitim programı, microservices mimarisinin temelleriyle başlar. Katılımcılar, veri yönetimi, mesajlaşma sistemlerinin yönetimi ve hata toleransı gibi temel kavramları öğrenirler. Ayrıca, microservices tabanlı modern uygulamaların tasarlanması ve yönetilmesi konusunda nasıl bir rol oynadığına dair bilgi sahibi olurlar. Bu bilgiler, katılımcıların uygulama tasarımı ve geliştirme sürecinde ihtiyaç duyacakları temel yapı taşlarını oluşturur.

Eğitimde, microservices mimarisi ve bununla ilişkili temel özellikler ve bileşenler üzerinde duruyoruz. Bu, katılımcılara real-time veri işleme, büyük veri akışları ve hata toleransı gibi yetenekleri kazandırır. Konu akışları, veri yönetimi ve dağıtık sistemlerin yönetimi gibi konular işlenir.

Son olarak, bir uygulamanın nasıl microservices mimarisi tabanlı geliştirileceği hakkında bilgi veriyoruz. Bu süreç, uygulamanın son testlerini yapmayı, veri yönetimini, ve en sonunda uygulamanın microservices tabanlı geliştirilmesini içerir. Bu bilgiler, katılımcıların uygulamalarını başarılı bir şekilde microservices tabanlı tasarlamalarına ve geliştirmelerine yardımcı olur.

Neler Öğreneceksiniz?

Microservices Mimarisi Eğitiminde katılımcılar;

- Mikro hizmetler kavramı: Mikro hizmetler nedir, nasıl çalışır ve avantajları nelerdir gibi konuları öğreneceklerdir.
- Mikro hizmetler mimarisi: Mikro hizmetler mimarisinin tasarımı, uygulanması ve yönetimi gibi konular ele alınacaktır.

- Veri akışı ve haberleşme: Mikro hizmetler arasındaki veri akışı ve haberleşme mekanizmalarını öğreneceklerdir.
- Performans ve güvenilirlik: Mikro hizmetler mimarisinde performans ve güvenilirlik sorunlarının nasıl çözüleceği gibi konular ele alınacaktır.
- Uygulama örnekleri: Mikro hizmetler mimarisinin uygulama örnekleri incelenebilir ve bu örnekler üzerinde uygulama yapılacaktır.
- Büyük veri ve bulut tabanlı sistemler: Mikro hizmetler mimarisinin büyük veri ve bulut tabanlı sistemlerle nasıl entegre edileceği gibi konular ele alınacaktır.

Ön Koşullar

Microservices Mimarisi Eğitiminin ön koşulları:

- Programlama dilleri: .Net, Java, Python veya benzer bir programlama dili gibi bir programlama dili bilmek gereklidir.
- Web uygulamaları ve RESTful hizmetler: Web uygulamalarının nasıl oluşturulduğu ve RESTful hizmetlerin nasıl kullanıldığı gibi konular bilinmelidir.
- Veritabanı teknolojileri: SQL veya NoSQL gibi bir veritabanı teknolojisi bilmek gereklidir.
- Bulut tabanlı teknolojiler: Amazon Web Services (AWS), Microsoft Azure gibi bulut tabanlı teknolojiler ile çalışma deneyimi olması faydalıdır.

Kimler Katılmalı

Microservices Mimarisi Eğitimine aşağıdaki profesyonel gruplar katılabilir:

- Yazılım Geliştiricileri: Microservices mimarisi ile uygulama geliştirme ve dağıtma süreçlerini anlamak isteyen yazılım geliştiricileri.
- DevOps: Microservices mimarisi ile bulut tabanlı teknolojiler ve dağıtılmış sistemlerin nasıl yönetildiğini anlamak isteyen DevOps profesyonelleri.
- Ar-Ge ve İnovasyon: Microservices mimarisi ile fikirlerinin nasıl gerçekleştirilebileceğini anlamak isteyen Ar-Ge ve İnovasyon profesyonelleri.
- IT Yöneticileri: Mikroservis mimarisi ile uygulamaların nasıl tasarlandığını, nasıl yönetildiğini ve nasıl iyileştirilebileceğini anlamak isteyen IT yöneticileri.

Outline

Breaking Up Monoliths – Pros and Cons

- Traditional Monolithic Applications and Their Place
- Disadvantages of Monoliths
- Developer’s Woes
- Architecture Modernization
- Architecture Modernization Challenges

- Microservices Architecture is Not a Silver Bullet!
- What May Help?
- In-Class Discussion
- Summary

Microservices

- What is a “Microservice”?
- Unix Analogy
- Principles of Microservices
- Services within an SOA vs Microservices
- Properties and Attributes of Microservices
- Benefits of Using Microservices
- The Two-Pizza Teams
- Beware of Microservices Cons
- Anti-Pattern: Nanoservices
- The Twelve-Factor App Methodology
- The Select Factors
- Serverless Computing
- Microservices – Operational Aspects
- Summary

Microservices Architecture Defined

- The Microservices Architecture
- SOA Promises and Expectations
- Microservices Architecture vs SOA
- The ESB Connection
- Microservices Architecture Benefits
- Microservices Architecture Choices and Attributes
- Example: On-Line Banking Solution Based on MsA
- Distributed Computing Challenges
- Replaceable Component Architecture
- The Actor Model
- MapReduce Distributed Computing Framework
- Hadoop’s MapReduce Word Count Job Example
- What Can Make a Microservices Architecture Brittle?
- 4+1 Architectural View Model
- Summary

Containerization Systems for Microservices

- Infrastructure as Code
- Why Not Just Deploy My Code Manually?
- What is Docker
- Docker Containers vs Traditional Virtualization
- Docker is a Platform-as-a-Service
- Docker Integration
- Docker Services
- Docker Application Container Public Repository
- Container Registries
- Your Own Docker Image Registry

- Starting, Inspecting, and Stopping Docker Containers
- One Process per Container
- The Dockerfile
- Kubernetes
- What is OpenShift
- Summary

Commonly Used Patterns

- Why Use Patterns?
- Performance-Related Patterns
- More Performance-Related Patterns
- Pagination vs. Infinite Scrolling – UX Lazy Loading
- Integration Patterns
- More Integration Patterns
- The Service Mesh Integration Pattern
- Mesh Pros and Cons
- Service-to-Service Communication with Mesh
- Resilience-Related Patterns
- Summary

API Management

- API Management Defined
- The Traditional Point-to-point Integration Example
- It Raises Some Questions …
- The Facade Design Pattern
- API Management Conceptual Diagram
- Complimentary Services for Microservices
- What Else is Needed?
- The Driving Forces
- API Management Offerings
- The Mashery API Management System Overview
- AWS API Gateway Call Flow
- Summary

Designing and Implementing Microservices

- Two Types of IT Projects
- What is In Scope for a Robust Microservices Design?
- Scoping Your Microservice via the Bounded Context
- Scoping Your Solution's Microservices Architecture
- External / Shared and Internal Service Models
- General Architectural and Software Process Organizational Principles
- Loose Coupling, the OOD Perspective
- Crossing Process Boundary is Expensive!
- Cross Cutting Concerns
- More Cross Cutting Concerns
- To Centralize or Decentralize Client Access?
- Decentralized Client Access
- Centralized Client Access
- The Facade Pattern

- The Facade Service Conceptual Diagram
- The Naked Objects Architectural Pattern
- When to Use Naked Objects Pattern
- Dealing with the State
- How Can I Maintain State?
- Micro Front-ends (a.k.a. MicroUI)
- How can MicroUI Help Me?
- Your Clients Are Diverse
- The “Rich Client” – “Thin Server” Paradigm
- The “Rich Client” – “Thin Server” Architecture
- RIA as a Driving Force to Turn the “Thin Server” into a Set of Microservices
- Design for Failure
- Managing Failures Effectively
- The Immutable Infrastructure Principle
- Implementing Microservices
- JAX-RS
- Microservice-Oriented Application Frameworks and Platforms
- Embedding Databases
- Embedded Java Databases
- Summary

Microservices Integration

- One Common Observation
- The “One Service – One Host” Deployment
- Things to Consider when Integrating
- Technology Options
- The Data Exchange Interoperability Options
- The Correlation ID
- Enterprise Integration Patterns
- Asynchronous Communication
- Benefits of Message-Oriented Middleware (MOM)
- Asynchronous Communication Models
- Message Brokers
- A Message Broker Diagram
- Asynchronous Message Consumption Patterns
- Popular Messaging Systems
- Challenges of Managing Microservices
- Options for Managing Microservices
- In-Class Discussion
- Summary

Working with Data in Microservices

- Monolithic Databases
- The Traditional Two-phase Commit (2PC) Protocol
- Table Sharding and Partitioning
- The CAP Theorem
- Mechanisms to Guarantee a Single CAP Property
- The CAP Triangle

- Eventual Consistency
- Handling Transactions in Microservices Architecture
- The Event-Driven Data Sharing Diagram
- The Saga Pattern
- The Saga Log and Execution Coordinator
- The Saga Happy Path
- A Saga Compensatory Request Example
- In-Class Discussion
- The Need for Micro Databases
- Migrating Data from Existing Databases (Breaking up the Monolith Database)
- One Data Migration Approach
- One Data Migration Approach (Cont'd)
- In-Class Discussion
- Command Query Responsibility Segregation (CQRS)
- The CQRS Communications Diagram
- A Word of Caution
- The Event Sourcing Pattern
- Event Sourcing Example
- Applying Efficiencies to Event Sourcing
- Summary

Robust Microservices

- What Can Make a Microservices Architecture Brittle?
- Making it Resilient – Mechanisms
- Techniques and Patterns for Making Your Microservices Robust
- Fail Fast or Quiesce?
- Synchronous Communication Timeouts / Retries
- Asynchronous Communication Timeouts / Retries
- In-Class Discussion
- The Circuit Breaker Pattern
- The Circuit Breaker Pattern Diagram
- The Bulkhead Pattern
- Factor IX of the 12 App Methodology
- Feature Enablement
- Designing for Test and Failure
- Making Microservices Testable
- Test for Failure
- Continuous Testing and Integration
- Continuous Release and Deployment
- SLAs
- Where and What to Monitor
- Logging and Monitoring
- Summary