Building Microservices

Duration: 5 Days

Day 1: Introduction to Microservices & First Service

Module 1: Introduction to Microservices

- Microservices vs Monolithic Architecture
- Core principles of microservices
- Real-world use cases

Module 2: Your First Microservice

- Setting up the .NET development environment
- Creating a basic ASP.NET Core Web API
- Solution and project structure for microservices

Module 3: Adding Database Storage

- Introduction to relational databases
- Setting up SQL Server or SQLite
- Using Entity Framework Core for data access

Lab: Build and run a microservice with database integration (e.g., Product Service).

Day 2: Expanding the System & Communication Basics

Module 4: Preparing for the Next Microservice

- Creating additional microservices (e.g., Order Service)
- Independent deployments and service isolation

Module 5: Synchronous Inter-Service Communication

- REST API integration using HttpClient
- Handling timeouts and retries

Module 6: Asynchronous Inter-Service Communication

- Basics of messaging systems (RabbitMQ or similar)
- Implementing event-based communication

Lab: Implement inter-service communication using both HTTP and asynchronous messaging.

Day 3: Frontend and Microservice Security Introduction

Module 7: Initial Frontend Integration

- Building a simple frontend using HTML/JavaScript or Blazor
- Calling microservices APIs from frontend

Module 8: Identity in Microservices

- Basics of authentication and identity providers
- Understanding tokens (JWT) and claims

Module 9: Microservices Security Patterns and Techniques

- Gateway authentication
- Service-to-service token validation
- Secure-by-default practices

Lab: Connect a frontend to microservices with secure token-based authentication.

Day 4: Security Implementation and Authorization

Module 10: Implementing Microservices Security

- Enabling authentication middleware in ASP.NET Core
- Securing API endpoints using JWT tokens

Module 11: Authorization in Microservices

- Role-based and policy-based access control
- Using claims for custom authorization logic

Module 12: Frontend Integration to Secure Microservices

- Managing login and tokens in frontend
- Handling authorization failures and session expiry

Lab: Secure multiple services and integrate authenticated access in the frontend.

Day 5: Transactions and Store Experience

Module 13: Transactions with Microservices

- Challenges with distributed transactions
- Eventual consistency and reliability concerns

Module 14: Preparing the Saga Participants

- Saga design pattern fundamentals
- Setting up participant services (Order, Payment, Inventory)

Module 15: Implementing the Purchase Saga

- Orchestration-based implementation
- Handling success/failure paths

Module 16: Compensation and Idempotency

- Undoing actions when part of a flow fails
- Ensuring operations are idempotent

Module 17: The Frontend Store Experience

- Integrating complete purchase flow
- Displaying products, placing orders, handling responses

Final Lab: Full store experience including secure login, order placement, and recovery.