[Formations]

Microservices avec Domain-Driven Design

3 jours	**Méthodes pédagogiques :** 60% théorique, 40% pratique à travers des ateliers de mise en pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

- **Définir une architecture basée sur les microservices**
- **Maîtriser les concepts essentiels du Domain-Driven Design (DDD)**
- **Apprendre à modéliser et à définir le périmètre d'un microservice**
- **Identifier les problématiques et les patterns de résolution d'une architecture microservices**
- **Définir une trajectoire de migration d'un existant vers une approche microservices**

L'architecture microservice est une approche attractive pour construire et faire évoluer des systèmes à grande échelle, avec un grand nombre d'utilisateurs ou avec un grand nombre d'équipes de développeurs qui doivent pouvoir travailler en relative autonomie pour livrer des fonctionnalités efficacement et indépendamment.

En matière de microservices, le focus est souvent sur les technologies, alors que les points clés sont la définition pertinente des contours des services. Ce n'est pas facile et c'est précisément pour cela que l'approche Domain-Driven Design est essentielle pour guider le découpage. Il s'agit aussi de penser en terme de système en soignant tout ce qui est nécessaire entre les services.

Cette formation couvre à la fois la théorie et les applications dans une présentation unifiée, avec des exercices pratiques. Elle présente les concepts essentiels de DDD, les techniques clés d'architecture ainsi que les principales technologies nécessaires pour les mettre en oeuvre. Et puisque cette mise en oeuvre commence le plus souvent sur des systèmes existants, cette formation décrit aussi des trajectoires de migration et de cohabitation avec le legacy.

Cette formation s'appuie sur notre expertise théorique sur le sujet et sur nos expériences d'accompagnement de nos clients sur ces architectures microservices. En particulier, nous décrivons aussi des modes d'échec classiques que nous avons observés.

Programme :

Faire le point sur les différentes définitions et enjeux des microservices

- API et approches API-first
- Autres façons de regarder les microservices : « damage control »: size, time to replace, decoupling, scalabilité, SOLID
- Polyglot persistence, polyglot services

Comprendre l'apport de Domain-Driven Design pour aider à définir les frontières des services

- Comprendre et identifier les Bounded Contexts
- Découvrir des notions de Context Mapping
- Réaliser les conséquences en terme de duplication (DRY vs Coupling) et de cohérence à terme (Eventual Consistency)

Comprendre une architecture microservices Architecture

- Comprendre la différence entre Choreography et Orchestration
- Intégration Asynchrone avec Event-Driven Architecture, messaging et patterns d'intégration EIP
- Intégration Synchrone et les conséquences
- Réaliser l'importance de la robustesse des contrats entre services, leur gouvernance pour les enjeux de compatibilité, les heuristiques et pièges à éviter
- Compromis Autonomy vs. Authority
- CQRS, Backend for Front-End (UI-specific API vs service API), mashup techniques

Penser "Design for Failure"

- Penser en terme de zones d'échec (Failure Areas)
- Circuit breakers for graceful degradation
- Découvrir les approches de test : compile time (Consumer-Driven Contracts) or runtime (Symian Army)
- When data get out of sync : reconciliation, replay, reload, et l'importance de l'idempotence et du polling comme solutions
- Monitoring : guidelines, outils, Alerting & Distributed tracing

Implémenter un service

- Concevoir une API, au-delà de la capacité "d'exposer les entités en REST"
- Persistence privative comme source de vérité (Sources of Truth), pattern Data Pump
- L'importance d'être Stateless, pour la fault-tolerance, le Cloud et l'elasticity
- Architecture Hexagonale
- Event-Sourcing vs Event-Driven
- Découvrir des patterns spécifiques pour le Cloud, la notions de Serverless & de Function as a Service
- Microservice Chassis: Spring Boot, Dropwizard
- Horizontal Scaling, Load balancing and intelligent routing
- Configuration services, Registration and Discovery
- Principes de sécurité

Migration Legacy

- Requalifier un existant legacy en "Macroservices"
- Extension d'un existant par des nouveaux services dans une approche de fédération
- Le pattern clé Strangler Application et autres patterns legacy utiles : Legacy Read Model, Double Feeding, Services Aggregation, Change Data Capture
- Les enjeux des périmètres transactionnels historiques : Save on Out of Focus vs. Distributed Transaction vs. Large microservices

Introduire une architecture microservices et les principaux pièges à éviter

- A quel moment et comment adopter les microservices
- Un changement de culture au-delà d'un changement technique
- Alternatives aux microservices : modularité, Discipline, approche microservice-Ready, et les limites associées
- Compétences et maturité pré-requises

Public :

Cette formation s'adresse aux développeurs expérimentés, aux architectes, team leads, référents et coachs techniques.

Pré-requis :

- Bonne connaissance d'au moins un langage de programmation.
- Les participants doivent avoir un poste de travail avec un environnement de développement.
- Avoir déjà une expérience dans la conception logicielle dans le cadre de projets d'entreprise.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.