

## Formation Architecture Microservices avec Spring Boot

<b>Durée :</b>	5 jours
<b>Public :</b>	Développeurs Java EE
<b>Pré-requis :</b>	Maîtriser la programmation orientée objet en Java - Maîtrise Spring Core
<b>Objectifs :</b>	Maîtriser l'utilisation de Spring Boot pour la construction de web services
<b>Sanction :</b>	Attestation de fin de stage mentionnant le résultat des acquis
<b>Taux de retour à l'emploi:</b>	Aucune donnée disponible
<b>Référence:</b>	JAV100901-F
<b>Note de satisfaction des participants:</b>	4,72 / 5

### Comprendre l'architecture de microservices

- Architecture en couches : du monolythe au microservices
- Contraintes d'architecture des microservices
- Gestion de l'authentification centralisée dans une architecture microservices
- Intérêt d'une passerelle d'API
- Gestion centralisée des traces

### Développement de microservices avec Spring Boot

- Galaxie Spring : présentation, apports
- Spring Framework : inversion de contrôle et injection de dépendances
- Spring Boot : principe, fonctionnalités, pré-requis
- Création d'un projet : starters, gestion des dépendances, packaging, exécution, debug
- Gestion du rechargement à chaud
- Configuration du projet (.properties ou .yml) et utilisation de profils ou d'une configuration externe
- Configuration de Logback pour la gestion des logs (logback.xml)
- Organisation des couches du projet : controllers, services, repositories
- Intérêt d'une couche de DTOs, utilisation d'un mapper d'objets
- Implémentation de contrôleurs REST : mapping global ou spécifique, méthodes, types de retours, annotations jackson
- Gestion des paramètres de méthodes et du mapping
- Gestion du download
- Gestion de l'upload, configuration
- Gestion des services et des transactions associées
- Gestion du cross origin et restriction des domaines appelants
- Test de l'api REST avec Postman
- Ecriture de tâches asynchrones, planification
- Cache web

## **Atelier : Écriture de micro-services avec Spring web - Test des méthodes de services avec Postman ou autre**

### **Documenter une API REST**

Open API Specification (Swagger) : présentation, outil  
Utilisation de Spring Doc Open API UI  
Visualisation avec Swagger Editor  
Documentation du code Java, génération de javadoc

### **Atelier : Documentation de l'api**

### **Intercepter des requêtes et gérer les erreurs**

ControllerAdvice et gestion globale des exceptions  
Capture d'exceptions personnalisées (@ExceptionHandler)  
Intercepteurs de requêtes/réponses

### **Atelier : Gestion des exceptions et implémentation d'intercepteurs**

### **Appeler d'autres API REST (écriture de clients)**

RestTemplate : présentation, méthodes  
Ecriture de requêtes GET, POST, PUT, DELETE - utilisation de la méthode exchange()  
Gestion des paramètres et du corps de la requête  
Gestion des headers  
Gestion des réponses et utilisation d'object mappers

### **Atelier : Implémentation de clients Java pour un service REST**

### **Gérer efficacement la couche de persistance**

Spring Data JPA : apports, mise en place, configurations multiples  
Mapping des entités, relations  
Gestion de la concurrence : @Version, locking  
Ecriture de repositories : requêtes avancées JP-QL, SQL  
Repositories personnalisés  
Gestion du chargement des collections : lazy vs eager  
Configuration du cache : @Cacheable  
Mise en place d'une solution d'audit de tables (historique de modifications)

### **Atelier : Implémentation d'une couche complète de persistance - mise en place d'un cache**

### **Sécuriser un service web**

Gestion des données d'entête  
Gestion de la sécurité avec Spring Security  
Gestion des utilisateurs et des rôles

### **Atelier : Intégration de Spring Security**

### **Tester une application Spring Boot**

Stratégies de tests, types supportés  
Configuration de l'application  
Mocking des couches de l'application  
Tests auto-configurés  
Exécution et reporting

**Atelier : implémentation et exécution de tests**