DÉVELOPPEMENT BACKEND

Spring Boot : développer des API REST en Java

Apprenez à créer des API REST avec Spring Boot 4 : injection de dépendances, contrôleurs et DTO, Spring Data JPA, gestion des erreurs, OpenAPI, sécurité JWT, tests et déploiement avec Docker.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
5j (35h)	**1 955 € / participant** Inter-entreprise. Intra : sur devis, au plus 1 955 € / participant	**Intermédiaire**	**Français**	**1-10**	**Formation** Présentiel ou à distance

1 PUBLIC VISÉ

- Développeurs Java qui connaissent la programmation orientée objet et veulent créer des API REST avec Spring Boot
- Personnes en reconversion qui visent un poste de développeur Java back-end

2 PRÉREQUIS

- Écrire des programmes Java orientés objet avec collections, exceptions et streams
- Écrire des requêtes SQL avec des jointures
- Aucune connaissance de Spring n’est requise

3 OBJECTIFS PÉDAGOGIQUES

- Créer avec Spring Initializr un projet Spring Boot 4 (Java 25, Maven), le configurer et le lancer
- Déclarer des beans, les injecter par constructeur et adapter la configuration selon le profil (dev, test, prod)
- Exposer les opérations CRUD d’une ressource en REST avec les bons statuts HTTP, des DTO distincts et la validation des données
- Persister des entités avec Spring Data JPA : repositories, méthodes dérivées, requêtes JPQL, pagination et tri
- Relier des entités sans le problème des N+1 requêtes, et garantir la cohérence d’une opération avec @Transactional
- Centraliser la gestion des erreurs pour renvoyer des réponses JSON cohérentes, sans trace d’exécution exposée
- Documenter l’API avec OpenAPI et Swagger UI
- Sécuriser l’API avec Spring Security et des jetons JWT, et restreindre les endpoints selon les rôles
- Tester services et endpoints avec JUnit, Mockito et MockMvc
- Préparer l’application pour la production : profils, migrations Flyway, image Docker et Docker Compose

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Projet Spring Boot et injection de dépendances

Module 1 : Créer un projet Spring Boot (3h30)

Spring et Spring Boot (1h30)

- Spring Framework et Spring Boot : ce que Boot simplifie
- Auto-configuration et starters
- Spring Boot 4 et Java 25
- Spring Initializr : Maven, dépendances
- Exercice d'entraînement : générer et lancer un projet vide

Structure, configuration et premier endpoint (2h)

- Structure : src/main/java, src/main/resources, pom.xml ; @SpringBootApplication
- application.yml ; DevTools ; lire les journaux de démarrage
- Un premier @RestController
- TP évalué : projet d'API de bibliothèque configuré dans application.yml, avec un endpoint /livres qui renvoie une liste en JSON

Module 2 : Injection de dépendances et configuration (3h30)

Inversion de contrôle et beans (1h30)

- Le problème résolu : couplage fort, tests difficiles
- Le conteneur ; @Component, @Service, @Repository
- Injection par constructeur ; @Bean dans une classe @Configuration
- Exercice d'entraînement : service injecté dans un contrôleur

Portée, profils et propriétés (2h)

- Portées singleton et prototype ; @PostConstruct
- @ConfigurationProperties : lire des paramètres typés
- Profils : @Profile, application-dev.yml, application-prod.yml
- TP évalué : service de notification avec deux implémentations choisies selon le profil actif, paramètres lus avec @ConfigurationProperties

Jour 2 (7h) : API REST et Spring Data JPA

Module 3 : Contrôleurs REST et DTO (3h30)

Endpoints REST (1h30)

- @RestController ; @GetMapping, @PostMapping, @PutMapping, @PatchMapping, @DeleteMapping
- @PathVariable, @RequestParam, @RequestBody ; ResponseEntity
- Statuts HTTP : 200, 201, 204, 400, 404
- Exercice d'entraînement : CRUD d'une ressource gardée en mémoire

DTO et validation (2h)

- Pourquoi ne pas exposer les entités ; des records Java comme DTO
- Passer de l'entité au DTO et inversement
- Bean Validation : @NotBlank, @Size, @Email, @Min ; @Valid
- TP évalué : ressource Livre complète : DTO de création et de réponse en records, validation, statuts HTTP justes

Module 4 : Entités et Spring Data JPA (3h30)

Entités JPA (1h30)

- @Entity, @Id, @GeneratedValue, @Column, @Enumerated ; dates avec java.time
- La source de données ; ddl-auto en développement
- Dates de création et de modification automatiques (auditing)

- Exercice d’entraînement : entités Livre et Auteur

Repositories Spring Data (2h)

- JpaRepository ; méthodes dérivées
- @Query en JPQL et en SQL natif
- Pageable et Page ; projections
- TP évalué : brancher la ressource Livre sur la base : recherche par mot-clé, pagination et tri

Jour 3 (7h) : Relations, transactions, erreurs et documentation

Module 5 : Relations, service et transactions (3h30)

Relations entre entités (1h30)

- @ManyToOne et @OneToMany ; @ManyToMany ; @OneToOne
- mappedBy ; cascade ; orphanRemoval
- Exercice d’entraînement : relation Livre et Auteur

Chargement et transactions (2h)

- Chargement LAZY et EAGER ; le problème des N+1 requêtes
- JOIN FETCH et @EntityGraph
- La couche service ; @Transactional, readOnly, retour arrière automatique
- TP évalué : endpoints des livres avec auteur et catégories sans N+1 requêtes, et emprunt d’un livre dans une transaction annulée si le livre n’est plus disponible

Module 6 : Gestion des erreurs et documentation (3h30)

Gestion globale des erreurs (1h30)

- @RestControllerAdvice et @ExceptionHandler
- ProblemDetail : le format standard des réponses d’erreur
- Erreurs de validation ; ressource introuvable
- Exercice d’entraînement : renvoyer une erreur 404 au format ProblemDetail

Documentation OpenAPI (2h)

- springdoc-openapi et Swagger UI
- @Operation, @ApiResponse, @Schema
- Conventions REST : nommage des ressources, statuts
- TP évalué : gestionnaire global d’erreurs au format ProblemDetail et documentation Swagger complète de l’API

Jour 4 (7h) : Sécurité et tests

Module 7 : Spring Security et JWT (3h30)

Spring Security (1h30)

- Filtres et SecurityFilterChain ; requestMatchers
- Mots de passe hachés avec BCrypt
- CORS ; CSRF et API sans état
- Exercice d’entraînement : endpoints publics et endpoints protégés

JWT et rôles (2h)

- Structure d’un JWT : en-tête, contenu, signature
- Émettre un jeton à la connexion ; le valider avec le support JWT intégré à Spring Security

- Rôles ; @PreAuthorize ; récupérer l'utilisateur connecté
- Jetons de rafraîchissement : le principe
- TP évalué : /auth/register et /auth/login qui renvoient un JWT, endpoints d'administration réservés au rôle ADMIN

Module 8 : Tests (3h30)

Tests unitaires (1h30)

- JUnit : @Test, @BeforeEach ; assertions avec AssertJ
- Mockito : @Mock, when, verify
- Exercice d'entraînement : tester le service d'emprunt avec un repository simulé

Tests d'intégration (2h)

- @WebMvcTest et MockMvc ; @SpringBootTest
- @WithMockUser pour les endpoints sécurisés
- @DataJpaTest pour les repositories
- TP évalué : tests unitaires du service d'emprunt et tests MockMvc des endpoints CRUD, y compris un accès refusé sans jeton

Jour 5 (7h) : Déploiement et projet de synthèse

Module 9 : Préparer la production (3h)

Configuration de production (1h30)

- JAR exécutable ; profil de production ; secrets dans des variables d'environnement
- PostgreSQL ; migrations avec Flyway
- Journaux avec SLF4J ; état de santé avec Actuator
- Exercice d'entraînement : écrire la migration Flyway initiale

Image Docker et Docker Compose (1h30)

- Dockerfile multi-stage, image JRE légère
- Docker Compose avec PostgreSQL
- TP évalué : conteneuriser l'API avec Docker Compose et PostgreSQL, profil de production, migrations Flyway au démarrage

Module 10 : Projet de synthèse, évalué (4h)

Réalisation (2h30)

- Projet de synthèse, réalisé individuellement : API de bibliothèque complète : couches contrôleur, service et repository, entités reliées, pagination, JWT et rôles, DTO, erreurs au format ProblemDetail, tests du service principal, documentation Swagger, Docker Compose

Revue et retour (1h30)

- Revue de code collective : architecture, sécurité, tests
- Retour individuel du formateur sur le projet rendu

Le contenu peut être ajusté selon le niveau, les prérequis et les besoins des participants.

5 COMPÉTENCES VISÉES

- Créer et configurer une application Spring Boot
- Concevoir une API REST propre
- Gérer les données avec Spring Data JPA

- Sécuriser une API REST
- Tester et documenter une API
- Déployer une application Spring Boot

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou à distance, en classe virtuelle. Le formateur alterne entre méthode démonstrative, méthode interrogative (analyse des choix d'architecture et des conventions REST) et méthode active (exercices d'entraînement, TP évalués et projet de synthèse).

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec exercices et corrections par module
- Environnement de développement : IntelliJ IDEA (version gratuite) + JDK 25 + Docker, installés sur l'ordinateur du participant
- Pour la classe virtuelle : visioconférence (Microsoft Teams ou équivalent), partage d'écran, chat en direct
- Supports remis par e-mail et sur la plateforme pédagogique LaPolaris (ressources, émargement)

En formation intra-entreprise sur le site du client, celui-ci s'assure et s'engage à disposer de toutes les ressources matérielles nécessaires à son bon déroulement (salle, postes de travail, logiciels, accès à Internet), conformément aux prérequis indiqués dans le programme et aux moyens prévus dans la convention.

8 MODALITÉS D'ÉVALUATION

- Questionnaire de positionnement individuel renseigné par chaque participant avant le démarrage de la formation, analysé par le formateur et donnant lieu, le cas échéant, à un ajustement du déroulé
- Exercices d'entraînement réalisés par chaque participant sur son poste, corrigés en séance
- Évaluation des acquis par les TP évalués, réalisés individuellement sur poste : un par module, puis le projet de synthèse décrit dans le dernier module du programme ; chaque objectif est évalué dans au moins un TP
- Grille de suivi renseignée par le formateur pour chaque participant et pour chacun des objectifs de l'action, au regard des critères d'évaluation ci-dessous
- Questionnaire d'appréciation renseigné par les participants en clôture de formation, et enquête à froid dans un délai de trois à six mois

9 CRITÈRES D'ÉVALUATION

- Projet généré avec Spring Initializr ; configuration dans application.yml ; application qui démarre sans erreur
- Dépendances injectées par constructeur ; implémentation choisie selon le profil actif ; paramètres lus depuis la configuration
- Endpoints CRUD avec les bons verbes et statuts (201, 204, 404) ; DTO distincts des entités ; données invalides refusées avec un statut 400
- Entités persistées ; recherche par méthode dérivée ou JPQL ; pagination et tri renvoyés par l'API

- Relations déclarées des deux côtés ; liste chargée sans N+1 requêtes ; opération annulée entièrement en cas d’erreur
- Toutes les erreurs renvoient un JSON au même format ; aucune trace d’exécution dans les réponses
- Tous les endpoints documentés dans Swagger UI, avec leurs réponses possibles
- Connexion qui renvoie un JWT ; endpoints protégés refusés sans jeton valide ; endpoints d’administration réservés au rôle ADMIN
- Tests unitaires du service avec des mocks ; tests MockMvc des endpoints, cas d’erreur compris ; tous les tests passent
- Application lancée avec Docker Compose et PostgreSQL ; profil de production ; schéma créé par Flyway ; secrets dans des variables d’environnement

10 INDICATEURS DE RÉSULTATS

Les indicateurs de résultats de cette formation (taux de satisfaction et taux de réussite) seront publiés sur sa fiche en ligne à l’issue des premières sessions réalisées.

11 MODALITÉS DE VALIDATION

Une attestation de fin de formation est remise à chaque participant à l’issue du parcours. Elle mentionne **l’intitulé** et la **nature de l’action**, sa **durée** en heures, sa **période de réalisation**, **les objectifs de l’action** et le **niveau d’atteinte** de chacun d’eux, exprimé en trois valeurs : **atteint**, **partiellement atteint**, **non atteint**. **Une attestation d’assiduité**, établie à partir des émargements, est délivrée dans les mêmes conditions.

Formation non certifiante : elle ne délivre pas de certification enregistrée au RNCP ou au Répertoire spécifique.

12 SUIVI ET ACCOMPAGNEMENT

- Émargement électronique signé par chaque participant et par le formateur à chaque demi-journée, en présentiel comme à distance
- Traçabilité des activités pédagogiques réalisées
- Attestation d’assiduité délivrée en fin de formation
- Suivi individuel via les QCM de fin de journée, les exercices corrigés et les TP évalués

13 CONDITIONS D'ACCÈS

Formation accessible sans prérequis administratif particulier. Cette formation n’est pas éligible aux financements publics ou mutualisés, notamment au compte personnel de formation et à une prise en charge par un opérateur de compétences.

14 DÉLAIS D'ACCÈS

Inscription possible jusqu’à 15 jours ouvrés avant le début de la session pour permettre la préparation logistique et, si besoin, l’adaptation du contenu au contexte client. Pour toute demande spécifique, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations peuvent être accessibles aux personnes en situation de handicap. La question d'un besoin d'aménagement (matériel, temporel ou pédagogique) est posée lors de l'analyse du besoin et du questionnaire de positionnement ; vous pouvez aussi nous le signaler à tout moment, y compris pendant la formation, afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr