Thème : Construire une API REST professionnelle avec Spring Boot en appliquant les principes SOLID et l'architecture en couches

Description

Cette formation intensive de cinq (05) jours est conçue spécifiquement pour des étudiants en génie logiciel ayant déjà des bases en programmation objet (Java) et en algorithmique. L'objectif est de les plonger dans l'écosystème professionnel de Spring Boot en adoptant une approche architecturale plutôt que technique. Plutôt que d'énumérer des fonctionnalités, le programme met l'accent sur la compréhension des patterns d'architecture d'entreprise : inversion de contrôle, injection de dépendances, séparation des responsabilités, et structuration en couches.

Durant les cinq (05) jours, les étudiants construiront une API REST complète et professionnelle, de la conception à la dockerisation, en appliquant les principes SOLID enseignés dans leur cursus. Chaque jour alterne théorie conceptuelle et pratique immédiate, avec un fil conducteur : la création progressive d'une application de gestion de produits qui servira de référence pour leurs futurs projets.

À l'issue de cette formation, les étudiants disposeront d'une compétence immédiatement valorisable : la capacité à développer une API REST robuste avec Spring Boot, en respectant les standards de l'industrie (documentation OpenAPI, gestion des erreurs, tests, dockerisation).

Objectifs pédagogiques

- Maîtriser l'architecture en couches (Controller → Service → Repository) ;
- Comprendre l'inversion de contrôle et l'injection de dépendances ;
- Appliquer les principes SOLID dans un projet concret ;
- Créer un projet Spring Boot ;
- Iimplémenter un CRUD complet avec JPA/Hibernate ;
- Exposer une API REST conforme aux bonnes pratiques ;
- Gérer les exceptions de manière centralisée ;
- Valider les données d’entrée ;
- Utiliser Spring Initializr ;
- Manipuler Postman pour tester les API ;
- Configurer une base de données (H2 puis PostgreSQL) ;

- dockeriser l’application ;
- Documenter automatiquement une API avec OpenAPI ;
- Écrire des tests d'intégration ;
- Structurer un projet maintenable et évolutif

Public cible

- Étudiants en génie logiciel, informatique ou équivalent

Prérequis

- Techniques
    - Maîtrise des bases de Java (classes, objets, héritage, interfaces)
    - Notions d'algorithmique et de structures de données
    - Connaissances basiques en SQL (requêtes SELECT, INSERT, UPDATE, DELETE)
- Matériels
    - Ordinateur avec JDK 17, STS, Docker Desktop, Postman

Programme détaillé

Jour 1 : Découverte de l'écosystème Spring Boot

Objectif : Comprendre la philosophie de Spring Boot et créer une première API REST simple

Phase	Contenu
Théorie (1h)	• Pourquoi Spring Boot ? • Inversion de contrôle (IoC) et injection de dépendances ; • Structure d'un projet Spring Boot ; • Spring Initializr et choix des dépendances ; • Annotations fondamentales
Pratique (3h)	• Création du projet via Spring Initializr ; • Exploration de la structure ; • Création d'un contrôleur REST ; • Test avec Postman ; • Compréhension du fichier application.properties

Jour 2 : Couche Repository & JPA

Objectif : Connecter l'application à une base de données et maîtriser l'abstraction JPA

Phase	Contenu
Théorie (30 mn)	• Spring Data JPA : philosophie ; • Mapping objet-relationnel avec JPA ; • Pattern Repository et JpaRepository ; • Bases de données embarquées vs production (H2, PostgreSQL)
Pratique (3h30)	• Création d'une entité Product ; • Configuration de H2 avec console d'administration ; • Création d'un repository ; • Insertion de données initiales avec CommandLineRunner ; • Découverte des méthodes héritées et création de méthodes personnalisées

Jour 3 : Architecture en couches & Services

Objectif : Structurer l'application selon les principes SOLID et introduire les DTO

Phase	Contenu
Théorie (1h)	• Pourquoi une couche Service ? • Séparation des responsabilités ; • Injection de dépendances par constructeur ; • Gestion des transactions ; • DTO (Data Transfer Object) et mapping
Pratique (3h)	• Création du ProductService ; • Implémentation des méthodes CRUD ; • Création des DTO (ProductRequest, ProductResponse) • Mapping manuel puis avec MapStruct ; • Refactorisation du contrôleur pour appeler le service ; • Tests unitaires du service avec JUnit

Jour 4 : API REST complète & Gestion des erreurs

Objectif : Exposer une API professionnelle et robuste avec validation et exception handling

Phase	Contenu

Théorie (1h)	• Bonnes pratiques REST (verbes HTTP, codes de statut, endpoints nommés) ; • Gestion centralisée des exceptions ; • Validation des données avec Bean Validation ; • Pagination et tri avec Pageable
Pratique (3h)	• Implémentation complète du contrôleur ; • Ajout de la validation sur les requêtes ; • Création d'un gestionnaire global d'exceptionr ; • Gestion des exceptions métier personnalisées ; • Ajout de la pagination sur la liste des produits ; • Tests exhaustifs avec Postman

Jour 5 : Documentation & Mise en production

Objectif : Produire une API documentée, testée et prête pour le déploiement

Phase	Contenu
Théorie (1h)	• Documentation automatique avec SpringDoc OpenAPI (Swagger) ; • Configuration multi-environnements ; • Tests d'intégration avec @SpringBootTest ; • Mise en production (jar exécutable, Docker)
Pratique (3h)	• Intégration de SpringDoc OpenAPI ; • Accès à l'interface /swagger-ui.html ; • Création d'un test d'intégration pour un endpoint ; • Configuration pour environnement de production ; • Création d'un Dockerfile et build de l'image ; • Optionnel : déploiement sur Railway/Render

Ressources fournies

Type	Contenu
Dépôt GitHub	• Projet squelette avec branches par jour
Documentation	• Fiche récapitulative des annotations essentielles ; • Guide de résolution des erreurs courantes
Outils	• Collection Postman prête à l'emploi ; • Scripts SQL d'initialisation ; • Dockerfile et docker-compose de référence