Bienvenue à ProSkills IT – Formations professionnelles au Togo

Fiche du cours

50 h

Titre :

Web300 - Java Spring Boot Backend

Description :

Construire une API REST prête pour la production avec Java 21 (LTS) et Spring Boot 3.x : controllers & DTO, validation Jakarta et Problem Details, JPA/Hibernate (PostgreSQL) avec stratégies de performance, migrations Flyway (baseline + repeatables), Spring Security 6 (JWT stateless, access/refresh, rotation de clés), documentation OpenAPI/Swagger, tests (JUnit 5/Mockito/Spring Boot Test + Testcontainers requis), observabilité (Actuator, Micrometer/Prometheus, logs JSON corrélés, aperçu OpenTelemetry), résilience (Resilience4j : retry/circuit breaker, rate-limiting), Docker et CI GitHub Actions. Cours 100% pratique avec un mini-produit fil rouge (ex. Catalogue & Commandes).

Objectifs :

- Setup & structure : Spring Initializr, Maven/Gradle, packaging “hexagonal”, config externalisée (profils dev/prod, variables d’env, secrets).*
- API & contrats : controllers/DTO REST, Jakarta Validation, Problem Details, OpenAPI/Swagger (exemples, codes) + export Postman.*
- Données & schéma : Spring Data JPA (UUID, relations, pagination/transactions) avec perfs (anti-N+1 via projections/entity graphs/fetch plan) + Flyway (baseline & repeatables).*
- Sécurité : Spring Security 6 avec JWT (access/refresh), rôles/authorities, CORS piloté par l’env, notions de rotation de clés et bonnes pratiques CSRF côté API.*
- Qualité, CI & conteneurs : tests JUnit 5/Mockito/Spring Boot Test + Testcontainers (Postgres) obligatoires en local & CI ; Docker (multi-stage/Jib) et CI matrix (Java 17/21).*
- Observabilité & résilience : Actuator, Micrometer/Prometheus, logs JSON corrélés (traceId), aperçu OpenTelemetry ; Resilience4j (retry/circuit breaker/bulkhead), rate limiting et idempotence.

Chapitres :

1. Setup & Architecture — Java 21, Spring Boot 3.x, Initializr, Maven/Gradle • structure hexagonale • profils & config externalisée (.env/secrets) • conventions Git/commit.*
2. REST & Erreurs — controllers, DTO, Jakarta Validation (@Valid) • handler global & Problem Details • pagination/tri/conventions de réponse.*

3. JPA/Hibernate I — entités UUID, relations (@OneToMany/@ManyToOne), repositories, pagination & sorting, contraintes & intégrité.*

4. JPA/Hibernate II & Flyway — Flyway (baseline, repeatables) • requêtes dérivées & @Query • projections, entity graphs, fetch plan, transactions • élimination N+1.*

5. Sécurité (Spring Security 6) — chaîne stateless, JWT access/refresh (notions de rotation HS→RS), rôles/authorities • CORS par env, durcissement basique, CSRF & API.*

6. Contrats & Doc — springdoc-openapi (schemas, exemples, codes) • versioning /api/v1 • export Postman & DX.*

7. Tests & Qualité — JUnit 5, Mockito, Spring Boot Test (slices) • Testcontainers Postgres (obligatoire local & CI) • couverture & règles de qualité.*

8. Observabilité & Résilience — Actuator, Micrometer/Prometheus, logs JSON avec traceId • santé/readiness • Resilience4j (retry/circuit breaker/bulkhead), rate limiting simple, idempotence.*

9. Docker & CI — Dockerfile multi-stage / Jib, docker-compose (app + Postgres) • CI GitHub Actions (matrix Java 17/21, tests + Testcontainers, build & push image).*

10. Capstone — API Catalogue & Commandes : sécurité JWT, JPA optimisée (sans N+1), doc OpenAPI soignée, tests d'intégration Testcontainers, métriques & logs, Docker + CI • README pro & démo

À la fin :

Vous saurez livrer une API Spring Boot réellement "prod-ready" : modèle JPA robuste (sans N+1), migrations Flyway propres, contrats OpenAPI soignés, sécurité JWT stateless, tests d'intégration réalistes (Testcontainers), observabilité complète (Actuator, Micrometer/Prometheus, logs JSON corrélés) et résilience (Resilience4j, rate-limit). Vous remettrez un dépôt dockerisé avec CI fonctionnelle (build + tests + image), README pro