Fiche du cours

60 h

Titre :

SE201 - Génie logiciel 2 (SOLID, Patterns, Architecture, Logging)

Description :

On passe au niveau architecture : SOLID pour la maintenabilité, design patterns (Factory, Strategy, Observer + Adapter/Template Method en bonus), architecture hexagonale (ports & adapters), logging structuré, configuration .env, et bases d'observabilité (logs/metrics/traces). Tronc commun multi-langages avec exemples en Python et miroirs C++/Java. Prérequis : SE-101.

Objectifs :

- Appliquer SOLID pour réduire le couplage et faciliter l'évolution.*
- Implémenter Factory, Strategy, Observer (+ Adapter/Template Method si temps) sur un mini-domaine.*
- Structurer une architecture hexagonale (Domain / Ports / Adapters) prête aux tests.*
- Mettre en place logging JSON, gestion d'erreurs et config propre (12-Factor).*
- Comprendre observabilité (logs/metrics/traces) et résilience (retry, timeouts, circuit-breaker – aperçu).

Chapitres :

1. SOLID (SRP/OCP/LSP/ISP/DIP) : exemples & anti-exemples*
2. SOLID — atelier refactoring guidé par tests*
3. Patterns 1 : Factory, Strategy (variantes), cas concrets*
4. Patterns 2 : Observer (+ Adapter/Template Method en bonus)*
5. Architecture hexagonale : séparation Domain / Ports / Adapters*
6. Hexa — atelier : adapter une app "spaghetti" en ports/adapters*
7. Logging structuré (JSON), niveaux/loggers, propagation d'erreurs*
8. Configuration & secrets : .env, variables d'environnement, 12-Factor*
9. Observabilité : logs/metrics/traces (OpenTelemetry – aperçu), résilience (retry/timeout)*
10. Mini-projet SE-201 : micro-service "hexagonal" avec logs, config .env, tests, et guide d'exécution

À la fin :

Vous serez capable de concevoir et faire évoluer une application selon des principes d’architecture sains, d’appliquer des patterns adaptés, de journaliser utilement et de gérer la configuration proprement — des compétences transférables sur Python, C++ et Java.