

Formation Pragmatic Architecture : concevoir des systèmes pragmatiques et modulaires

Réf. SF34601

Maîtrisez les patterns, styles et stratégies d'architecture pour concevoir, moderniser et sécuriser vos systèmes d'entreprise.

2 j Durée	2 100 € HT Inter / participant	4 100 € HT Intra (≤12 pers.)	0 Sessions à venir
--------------	-----------------------------------	---------------------------------	-----------------------

Objectifs pédagogiques

- ✓ Expliquer les raisonnements et perspectives de l'architecture logicielle
- ✓ Diagnostiquer les problématiques techniques d'une architecture en production (messaging, consistance, monitoring)
- ✓ Appliquer les principes de modularité pour structurer un système en contextes autonomes
- ✓ Concevoir une stratégie de migration progressive d'un système existant
- ✓ Choisir le style architectural adapté à un contexte donné (hexagonal, event-driven, microservices)
- ✓ Intégrer la sécurité dès la conception conformément au Cyber Resilience Act

Public visé & prérequis

Public visé Architectes, Développeurs

Prérequis- Maîtrise de la programmation orientée objet et du design logiciel - Expérience significative en développement logiciel (participation à un ou plusieurs projets) - Familiarité avec les concepts REST / HTTP / API

Programme détaillé

Raisonnement architectural et positionnement du rôle d'architecte

- Définitions multiples de l'architecture logicielle : stabilité, échelle, savoir partagé et décisions irréversibles
- Alignement business/IT : l'architecture comme levier d'alignement entre objectifs métier et systèmes d'information
- Perspectives d'architecture : architecture globale vs. locale, diagrammes Umbrella pour visualiser le périmètre
- Architecture vs. Architecte : rôle, responsabilités et posture dans l'organisation
- Architecture émergente : laisser l'architecture se révéler par les décisions incrémentales plutôt que par un big design upfront

Activités pratiques :

- Analyse de décisions architecturales réelles : identifier lesquelles sont réversibles et lesquelles ne le sont pas
- Cartographie d'un système existant à l'aide de diagrammes Umbrella

Principes de modularité et découpage en contextes autonomes

- Bounded Contexts et Context Mapping : identifier les frontières métier pour structurer un système en modules cohérents
- Microservices vs. Modular Monolith : critères de choix et implications organisationnelles
- Contrats rétrocompatibles entre modules : garantir l'évolution indépendante des composants
- Federated vs. Centralized Data : arbitrer entre autonomie locale et cohérence globale des données
- DRY vs. Coupling : accepter la duplication pour réduire le couplage entre modules
- Eventual Consistency : conséquences de la distribution des données sur la cohérence

Activités pratiques :

- Exercice de découpage d'un système monolithique en Bounded Contexts
- Arbitrage centralisation vs. fédération des données sur un cas concret

Styles et patterns architecturaux

- Event-Driven Architecture : chorégraphie vs. orchestration, forces et faiblesses de chaque approche
- Architecture Hexagonale : isoler le domaine métier des dépendances techniques (ports & adapters)
- Microservices : Smart Endpoints & Dumb Pipes, principes de conception et pièges courants
- Enterprise Integration Patterns (EIP, Gregor Hohpe) : les patterns d'intégration de référence entre systèmes
- PoEAA (Patterns of Enterprise Application Architecture, Fowler) : les patterns fondamentaux des applications d'entreprise
- Design d'API : principes de conception d'interfaces entre composants et systèmes

Activités pratiques :

- Comparaison de styles architecturaux sur un même cas d'usage : choix et justification
- Conception d'une architecture event-driven avec arbitrage chorégraphie/orchestration

Architecture technique en production

- Messaging asynchrone : queues vs. topics, events vs. commands, patterns de communication inter-services
- Consistance des données : Golden Source, réplication, autonomie vs. autorité sur les données
- Tolérance aux pannes et équilibrage de charge : stratégies de résilience pour les systèmes distribués
- Stateless vs. stateful : conséquences sur la scalabilité, stickiness, affinité et gestion des sessions
- Configuration et découverte de services : patterns de discovery dans les architectures distribuées
- Observabilité : logging, tracing et monitoring — guidelines, outils et alerting
- Test-Driven Architecture : valider les décisions architecturales par les métriques et l'analyse statique
- Living Documentation : documenter l'architecture par les annotations de code et les diagrammes générés au build

Activités pratiques :

- Diagnostic d'une architecture existante : identifier les problèmes de consistance et de résilience
- Mise en place d'une stratégie d'observabilité (logging, tracing, alerting) sur un cas d'étude

Stratégies de migration et modernisation de l'existant

- Principes de cohabitation legacy : pourquoi éviter le Big Bang et privilégier les migrations progressives
- Strangler Application Pattern : remplacer progressivement un système existant par de nouveaux composants
- Legacy Read Model : découpler la lecture des données d'un système hérité pour préparer la migration
- Change Data Capture et Macroservice : patterns avancés pour intégrer ancien et nouveau monde
- Raisonnement ROI sur les décisions de modernisation : évaluer le rapport coût/bénéfice d'une migration

Activités pratiques :

- Planification d'une migration progressive avec le pattern Strangler sur un système legacy
- Analyse coût/bénéfice d'un scénario de modernisation

Architecture sécurisée et conformité réglementaire

- Security by Design : intégrer la sécurité dès la conception plutôt qu'en surcouches
- Cyber Resilience Act (CRA) : comprendre les obligations réglementaires européennes et leur impact sur l'architecture
- Authentification et autorisation des services : OAuth2, patterns A&A dans les architectures distribuées

Activités pratiques :

- Audit de conformité CRA sur une architecture existante
- Intégration de contraintes sécurité dans un design applicatif

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

Note moyenne des participants de la filière Développement et Ingénierie de Logiciels : 4,9/5 — 165 avis

Donatien W. — Formation dense mais très digeste avec beaucoup de points abordés. Les spécificités de chacun ont pu être abordées.

Jean-Baptiste M. — Le format 3 jours est une contrainte forte. A la sortie de cette formation, on attaque à peine la "vallée du doute" et on sent qu'il y a...

Votre formateur

Enzo S. — Formateur

Enzo intervient notamment sur les langages et écosystèmes backend, ainsi que sur le software craftsmanship et l'excellence du code. Son activité de formation remonte à 2018.

Ils nous font confiance

Cette année notre filière Architecture & Design Logiciel a accompagné plus de 50 entreprises dont :

SOPRA STERIA CIMPA SAS THALES GLOBAL SERVICES SAS ADEO SERVICES AUBAY LBC FRANCE ORANGE BUSINESS SQLI WORLDLINE FRANCE SAS ADEVINTA +40 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com