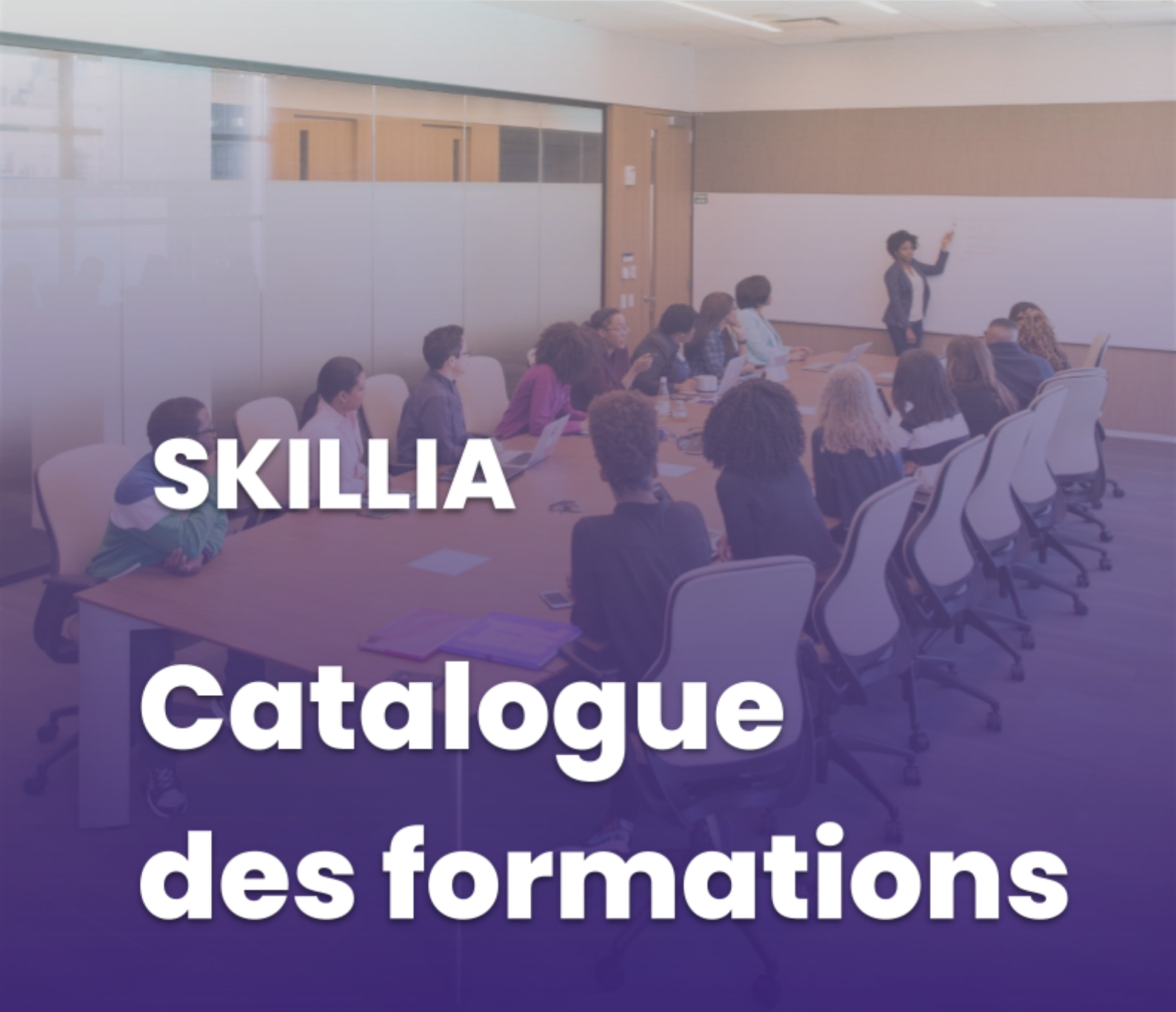


SKILLIA

Catalogue des formations

Innovez, Protégez, Transformez : les formations IA, Cyber et Digital signées Skillia

2025-2026

www.skillia.fr

Sommaire

Nos Valeurs

Une démarche axée sur la qualité

Chez Skillia, nous concevons des formations sur mesure, pensées pour répondre aux spécificités de votre organisation, de votre marché et de votre environnement.

La qualité constitue le pilier central de notre approche : pertinence des thématiques, sélection rigoureuse des experts, méthodes pédagogiques éprouvées, organisation soignée et mise à jour continue de notre offre pour anticiper les évolutions technologiques et les besoins du marché.

L'innovation au service de la formation

Créativité, innovation, écoute, réflexion et rigueur sont au cœur de notre ADN.

Ces valeurs guident chacune de nos actions pour vous offrir des formations à la pointe : des approches pédagogiques modernes, des contenus actualisés et une expérience d'apprentissage stimulante, adaptée aux enjeux d'aujourd'hui.

Une écoute active des entreprises

Chaque entreprise bénéficie d'un interlocuteur dédié, véritable consultant en formation.

Sa mission : analyser vos besoins, recommander les solutions les plus pertinentes en matière de contenu, d'organisation, de budget et de financement, puis vous accompagner tout au long du déploiement.

Il veille à la réussite du projet, à l'atteinte de vos objectifs et à la satisfaction des apprenants.

Pourquoi choisir Skillia ?

Des intervenants d’exception

es formateurs Skillia sont des experts reconnus, dotés de plusieurs années d’expérience dans leurs domaines respectifs et ayant occupé des postes à responsabilité en entreprise.

Ils conçoivent et animent eux-mêmes leurs formations, en s’appuyant sur une double expertise : une solide expérience de terrain et une maîtrise pédagogique éprouvée — véritable gage de qualité et de pertinence.

Une offre en constante évolution

Chez Skillia, l’innovation est continue. Nos programmes de formation sont régulièrement actualisés pour intégrer les dernières avancées technologiques et répondre aux nouveaux défis des entreprises.

Une approche résolument opérationnelle

es formations Skillia sont pensées pour une mise en pratique immédiate. Conçues et animées par des professionnels expérimentés, elles comportent au minimum 60 % d’exercices pratiques et d’études de cas inspirés de situations réelles. L’organisation progressive des modules permet d’acquérir les compétences nécessaires à chaque étape, dans les conditions les plus efficaces.

Une réactivité à chaque étape

Nos consultants en formation sont à votre écoute pour définir rapidement les solutions les plus adaptées à vos besoins : sélection de cours ou de parcours, formations intra ou sur mesure, ingénierie pédagogique et dispositifs d’accompagnement spécifiques à votre entreprise.

Former avant de recruter : un levier gagnant !

La reconversion professionnelle est une approche de recrutement innovante et efficace. Elle consiste à identifier des talents motivés, dotés du potentiel nécessaire, puis à leur offrir un parcours de formation sur mesure pour développer les compétences indispensables à leur futur métier.

Cette stratégie représente une opportunité mutuellement bénéfique : elle permet aux entreprises de répondre à leurs besoins en compétences tout en offrant aux candidats une véritable montée en qualification et une nouvelle trajectoire professionnelle.

Avantages pour l'entreprise

Combler le manque de compétences clés sur le marché

Recruter autrement : des profils compétents, motivés et alignés avec vos valeurs

Être en phase avec les dernières avancées technologiques

Avantages pour les candidats

Une reconversion professionnelle vers des métiers d'avenir

Valoriser son employabilité et répondre aux besoins réels du marché du travail.

Informations Pratiques SKILLIA :

Public visé : Professionnels (salariés, indépendants) et particuliers.

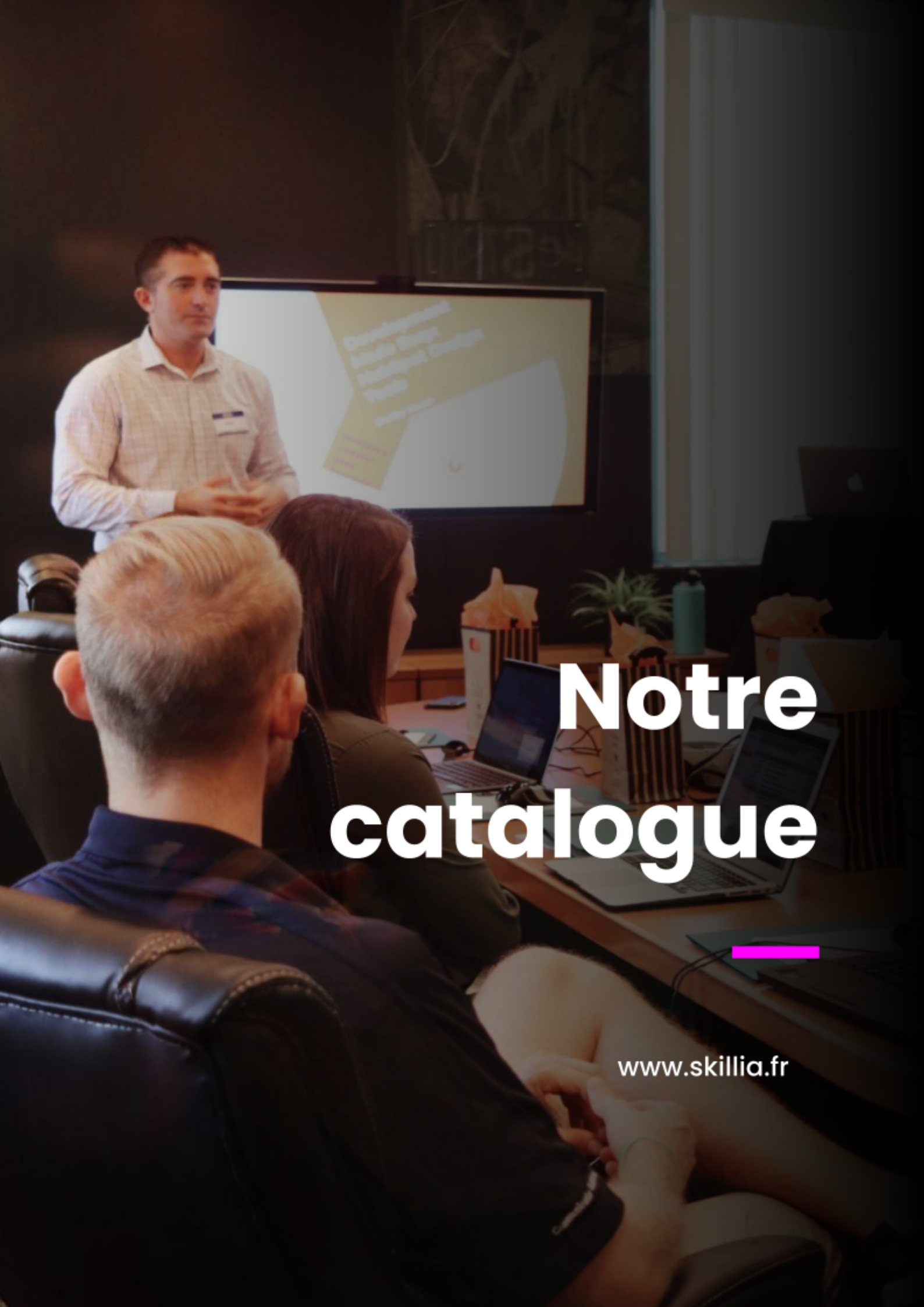
Modalités : Formations disponibles en **distanciel** (classe virtuelle synchrone) ou en **présentiel** (dans les locaux de nos partenaires ou en intra-entreprise).

Délais d'accès : Entre **15 et 30 jours** à compter de la signature de la convention et selon le mode de financement sollicité.

Accessibilité Handicap :

- **En distanciel :** Nos parcours sont adaptables (assistance technique, supports numériques accessibles).
- **En présentiel :** Nous nous assurons avec nos partenaires que les lieux d'accueil respectent les normes d'accessibilité (ERP).
- *Pour toute étude personnalisée de vos besoins spécifiques, contactez notre référent handicap :* ***Skander Maalej****.*

Contact : administration@skillia.fr



Notre catalogue

www.skillia.fr

Développement logiciel

Algorithmique – Maîtriser la logique de programmation

Prix : 1200 € H.T.

Durée : 2 jours

Public cible : Toute personne souhaitant acquérir les bases de la logique algorithmique avant d'aborder un langage de programmation ou de comprendre les bases des systèmes informatiques.

Prérequis : Aucun prérequis technique n'est nécessaire, mais une aisance avec l'ordinateur est recommandée.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre les fondements de l'algorithmique et des structures de contrôle
- Traduire un raisonnement logique en étapes claires (pseudo-code, organigrammes)
- Concevoir des algorithmes simples pour résoudre des problèmes
- Décomposer un problème en sous-tâches cohérentes
- Se préparer à l'apprentissage d'un langage de programmation (Python, JavaScript, etc.)

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à l'algorithmique

- Qu'est-ce qu'un algorithme ? Définition et exemples concrets
- Rôle de l'algorithmique dans l'informatique et la data
- Outils de modélisation : pseudo-code, logigramme, langage naturel

2. Structures de base

- Les types de données (entiers, chaînes, booléens)
- Variables et affectations
- Opérations arithmétiques et logiques

3. Contrôle du flux

- Structures conditionnelles : si, sinon, sinon si
- Boucles : pour, tant que
- Exercices de mise en pratique (calculs, analyse de chaînes, conversions)

4. Fonctions et modularité

- Définir une fonction, passer des arguments, retourner un résultat
- Réutilisation et lisibilité du code
- Décomposer un problème en sous-algorithmes

5. Initiation à la complexité

- Notion intuitive de performance algorithmique
- Meilleure solution vs solution fonctionnelle

6. Cas pratiques

- Algorithmes classiques : recherche, tri, maximum/minimum
- Résolution guidée de petits défis (fizzbuzz, palindrome, factorielle)
- Préparation à l'implémentation en Python (sans codage requis)

Structures de données et algorithmes avancés

Prix : 1900 € H.T.

Durée : 3 jours

Public cible : Développeurs débutants ou intermédiaires, analystes, techniciens ou toute personne souhaitant améliorer la performance et la structuration de son code.

Prérequis : Connaissances de base en algorithmique et en programmation (idéalement Python ou équivalent).

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Identifier et utiliser les structures de données adaptées à un problème donné
- Implémenter des algorithmes classiques et comprendre leur complexité
- Optimiser le traitement de données grâce à une meilleure structuration
- Approfondir leurs compétences pour aborder les domaines de la data science ou de l'IA

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Rappels fondamentaux

- Variables, types simples, fonctions
- Notion de complexité (temps/espaces, notation Big O)

2. Structures linéaires

- Listes, piles (stack) et files (queue)
- Implémentation, utilisation, complexité
- Cas d'usage : gestion d'historique, files d'attente

3. Structures hiérarchiques

- Arbres binaires : structure, parcours (préfixe, infixe, suffixe)
- Arbres de recherche (BST)
- Cas pratique : tri, organisation hiérarchique

4. Graphes

- Représentation (matrice d'adjacence, listes)
- Parcours en largeur (BFS) et en profondeur (DFS)
- Cas pratique : réseaux, chemins optimaux

5. Algorithmes de tri et de recherche

- Tri par sélection, insertion, bulle, rapide (quicksort), fusion (mergesort)
- Recherche linéaire, dichotomique
- Analyse de performance et choix adapté

6. Structures avancées

- Tables de hachage (dictionnaires)
- Heaps et files de priorité
- Cas d'usage : cache mémoire, planification

7. Mise en pratique

- Choisir la bonne structure pour un cas donné
- Optimisation d'un algorithme existant
- Résolution guidée de problèmes (format coding challenge)

Initiation à la programmation avec Python

Prix : 1910 € H.T.

Durée : 3 jours

Public cible : Toute personne souhaitant s'initier à la programmation à l'aide d'un langage simple, lisible et utilisé dans de nombreux domaines (automatisation, data, IA, web, etc.).

Prérequis : Aucun prérequis en programmation. Il est recommandé d'avoir suivi la formation « Algorithmique » ou d'en maîtriser les bases.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre le fonctionnement d'un programme écrit en Python
- Manipuler les types de données, structures de contrôle et fonctions
- Écrire des scripts simples pour automatiser des tâches ou résoudre des problèmes
- Lire et comprendre du code Python standard
- Se préparer à des applications avancées (data, IA, automation)

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à Python

- Historique, usages et philosophie du langage
- Installation de l'environnement (Python, IDE type VS Code ou Jupyter Notebook)
- Premiers scripts : structure d'un fichier Python, exécution

2. Types de base et opérations

- Nombres, chaînes de caractères, booléens
- Opérations arithmétiques, concaténation, comparaisons
- Conversion entre types

3. Structures de données

- Listes, tuples, dictionnaires, ensembles
- Parcours et manipulation d'éléments
- Fonctions intégrées utiles (len, range, enumerate, zip...)

4. Contrôle du flux

- Instructions conditionnelles (if, elif, else)
- Boucles for et while
- Gestion des exceptions (try, except)

5. Fonctions et modularité

- Définir une fonction avec ou sans paramètres
- Retourner une valeur, portée des variables
- Modules et importations

6. Introduction aux fichiers et à la console

- Lire et écrire des fichiers texte (.txt, .csv)
- Entrée utilisateur avec input(), impression avec print()

7. Petits projets guidés

- Calculatrice simple, conversion d’unités
- Analyse d’un fichier texte (ex : comptage de mots)
- Mini-jeu texte (ex : devine le nombre)

Manipuler et analyser des données avec Python (Pandas & NumPy)

Prix : 2200 € H.T.

Durée : 3 jours

Public cible : Développeurs, analystes, ingénieurs, data scientists débutants ou toute personne souhaitant apprendre à traiter efficacement des données tabulaires avec Python.

Prérequis : Bonne maîtrise des bases de Python (structures de données, fonctions, boucles). Une familiarité avec les fichiers CSV ou les bases de données est un plus.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Lire, nettoyer et transformer des données avec Pandas
- Réaliser des analyses statistiques simples
- Comprendre les bases du calcul vectoriel avec NumPy
- Préparer des données pour des visualisations ou des modèles IA

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à l'analyse de données

- La data dans les projets numériques : enjeux et étapes
- Présentation de l'écosystème Python pour la data (Pandas, NumPy, Matplotlib, etc.)

2. Manipulation de données avec Pandas

- Séries et DataFrames : création, lecture depuis CSV/Excel
- Accès, filtrage et sélection des données
- Opérations de nettoyage : traitement des valeurs manquantes, renommage, typage

3. Transformations et jointures

- Filtres conditionnels, transformations par colonnes
- Fusion de DataFrames (merge, concat)
- Agrégations et regroupements (groupby, pivot tables)

4. Introduction à NumPy

- Tableaux (ndarray), vecteurs et matrices
- Indexation, slicing, opérations vectorisées
- Performance vs boucles classiques

5. Statistiques descriptives

- Moyenne, médiane, écart-type, corrélation
- Identification des outliers et valeurs atypiques
- Tri croisé, tableaux récapitulatifs

6. Mise en pratique

- Étude de cas sur données réelles (CSV, Excel)
- Mini-projets guidés : analyse de ventes, scoring de clients, tendances temporelles

Visualiser des données avec Python (Matplotlib, Seaborn, Plotly)

Prix : 1460 € H.T.

Durée : 2 jours

Public cible : Professionnels manipulant des données (analystes, chargés d'études, chefs de projet, développeurs) souhaitant produire des visualisations claires et percutantes.

Prérequis : Maîtrise de base de Python et du package Pandas. Avoir suivi la formation "Manipuler et analyser des données avec Python" est recommandé.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Créer des graphiques avec les bibliothèques Python les plus courantes
- Adapter les types de graphiques aux types de données et aux messages à faire passer
- Personnaliser les visuels pour améliorer leur lisibilité et leur impact
- Réaliser des visualisations interactives avec Plotly

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à la visualisation de données

- Rôle de la data visualisation : comprendre, explorer, convaincre
- Choisir le bon graphique selon le message à transmettre
- Présentation des bibliothèques : Matplotlib, Seaborn, Plotly

2. Visualisation avec Matplotlib

- Création de courbes simples, barres, nuages de points
- Personnalisation : couleurs, légendes, titres, axes
- Gestion des sous-graphes (subplot)

3. Visualisation avancée avec Seaborn

- Graphiques statistiques (boxplot, violin, heatmap, pairplot…)
- Utilisation avec DataFrames Pandas
- Représentation multi-variables et catégorielles

4. Visualisations interactives avec Plotly

- Graphiques dynamiques : zoom, filtre, survol
- Intégration dans des notebooks ou dashboards simples
- Comparatif avec les outils statiques

5. Mise en pratique

- Analyse exploratoire guidée avec visualisations
- Construction d'un tableau de bord graphique
- Études de cas : indicateurs commerciaux, analyse de trafic, sondages

Programmation Orientée Objet (POO)

Prix : 1910 € H.T.

Durée : 3 jours

Public cible : Développeurs débutants ou ayant des connaissances en programmation (connaissance de base en Java ou d'un autre langage de programmation impératif)

Prérequis : Connaissances de base en programmation (structures de contrôle, variables, boucles, fonctions)

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre les principes fondamentaux de la programmation orientée objet
- Concevoir et développer des applications en utilisant la POO
- Implémenter des classes, objets, héritage, polymorphisme et encapsulation
- Appliquer les bonnes pratiques de la POO dans un environnement Java

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à la POO

- Définitions de la POO : Object, Class, Encapsulation, Héritage, Polymorphisme
- Historique et avantages de la POO
- Comparaison avec les langages procéduraux

2. Création d'une classe en Java

- Déclaration d'une classe et d'un objet
- Attributs, méthodes, et constructeurs
- Notion de portée (private, public, protected)

3. Encapsulation

- Accès aux attributs via les méthodes getters et setters
- Contrôle de l'accès aux données
- Avantages de l'encapsulation

4. Héritage

- Définition et mécanisme d'héritage (extends)
- Utilisation de la méthode super
- Surcharge et redéfinition des méthodes
- Avantages et limitations de l'héritage

5. Polymorphisme

- Surcharge de méthodes (overloading)
- Redéfinition de méthodes (overriding)
- Utilisation des interfaces pour le polymorphisme
- Exemple d'application du polymorphisme dans des projets réels

6. Abstraction

- Création de classes abstraites
- Définition d'interfaces et leur mise en œuvre
- Différence entre classe abstraite et interface

7. Exemples pratiques et études de cas

- Application des concepts dans des projets réels
- Études de cas sur la conception orientée objet
- Développement d'une mini-application en utilisant les principes de la POO

8. Bonnes pratiques en POO

- Respect des principes SOLID
- Utilisation d'annotations et gestion des exceptions
- Structuration du code et design patterns de base

Java pour Débutants

Prix : 3030 € H.T.

Durée : 5 jours

Public cible : Développeurs débutants ou toute personne souhaitant apprendre les bases du développement avec Java.

Prérequis : Aucune expérience préalable en programmation n'est nécessaire, mais une familiarité avec l'informatique de base (utilisation de systèmes d'exploitation, de logiciels) est recommandée.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre la syntaxe fondamentale du langage Java
- Créer des applications simples en Java en utilisant des concepts de programmation de base
- Manipuler les structures de données essentielles (tableaux, listes, etc.)
- Appliquer les bonnes pratiques de codage en Java
- Utiliser un environnement de développement Java moderne (JDK, IDE, Maven)

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à Java et à l'environnement de développement

- Historique et caractéristiques du langage Java
- Installation et configuration du JDK et d'un IDE (IntelliJ IDEA, Eclipse, etc.)
- Structure d'un programme Java : classes, méthodes, et la méthode main()
- Compilation et exécution d'un programme Java dans un IDE

2. Les bases du langage Java

- Types primitifs et variables (int, double, char, boolean)
- Utilisation des opérateurs arithmétiques et logiques
- Structures de contrôle : conditions (if, switch) et boucles (for, while, do-while)
- Tableaux : déclaration, initialisation et manipulation de tableaux simples
- Manipulation des chaînes de caractères (String), méthodes utiles

3. Introduction à la Programmation Orientée Objet (POO)

- Concepts clés : classes, objets, attributs et méthodes
- Création et instanciation de classes en Java
- Notion de constructor, gestion de la portée des variables (private, public, protected)
- Différence entre méthode statique et méthode d'instance

4. Gestion des erreurs et exceptions

- Concepts de base des exceptions en Java
- Bloc try/catch pour la gestion des erreurs
- Utilisation des exceptions personnalisées
- Bonnes pratiques de gestion des erreurs et exceptions en Java

5. Les collections et l'API standard de Java

- Introduction aux collections : List, Set, Map
- Création et manipulation de listes avec ArrayList, utilisation des itérateurs
- Manipulation des ensembles (HashSet, TreeSet) et des maps (HashMap, TreeMap)
- Tri des collections, utilisation des Comparator et Comparable

6. Travailler avec les fichiers et les entrées/sorties (I/O)

- Lecture et écriture dans des fichiers texte avec BufferedReader et BufferedWriter
- Sérialisation d'objets en Java
- Introduction aux flux de données en Java (fichiers texte, binaires)

7. Les dernières fonctionnalités de Java (version actuelle)

- Introduction à Java 17 et les fonctionnalités récentes :
- Pattern Matching pour instanceof (Java 16)
- Records pour simplifier la déclaration de classes de données (Java 14)
- Text Blocks pour faciliter la manipulation de chaînes multi-lignes (Java 13)
- Switch Expressions améliorées (Java 12)
- Sealed Classes pour restreindre les héritages (Java 17)
- Utilisation des nouvelles API : java.nio.file, java.util.stream, java.time (dates et heures)

8. Bonnes pratiques et structuration du code

- Respect des conventions de nommage
- Structuration d'un projet Java simple avec Maven
- Introduction à la gestion de versions avec Git
- Introduction aux tests unitaires avec JUnit

9. Exemples pratiques et exercices

- Développement d'un programme simple : gestion d'une bibliothèque, gestion des utilisateurs, ou calculatrice
- Résolution d'exercices pratiques pour appliquer les concepts appris

Java – Niveau Expert

Prix : 3030 € H.T.

Durée : 5 jours

Public cible : Développeurs expérimentés ayant une bonne maîtrise de Java et des concepts de programmation orientée objet, souhaitant approfondir leur expertise dans les domaines avancés du langage et des architectures Java complexes.

Prérequis : Maîtrise des bases de Java, des concepts fondamentaux de la POO, et des outils de développement Java (IDE, Maven). Expérience en développement avec Java ou tout autre langage orienté objet.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir et implémenter des architectures Java robustes, évolutives et performantes
- Utiliser les fonctionnalités avancées du langage Java pour résoudre des problèmes complexes
- Appliquer des principes de conception avancés, des design patterns, et des bonnes pratiques d'architecture logicielle
- Optimiser les performances d'applications Java en utilisant les dernières fonctionnalités et outils

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction aux concepts avancés de Java

- Vue d'ensemble de la plateforme Java et des évolutions récentes (Java 17 et au-delà)
- Comprendre l'architecture de la JVM, le garbage collector, et la gestion de la mémoire
- Analyse des dernières fonctionnalités du langage : Records, Pattern Matching, Sealed Classes, Text Blocks, etc.

2. Optimisation de la performance en Java

- Comprendre et analyser la gestion de la mémoire en Java : heap, stack, et garbage collection
- Optimisation des performances avec la JVM : tuning, monitoring (JVisualVM, JConsole, JProfiler)
- Stratégies de gestion de la mémoire et gestion des fuites mémoire
- Amélioration des performances avec des collections adaptées (ConcurrentHashMap, CopyOnWriteArrayList, etc.)
- Optimisation des entrées/sorties (I/O) : meilleures pratiques avec les flux, buffers, et NIO

3. Conception avancée avec la programmation fonctionnelle

- Lambda expressions et Stream API pour une programmation fonctionnelle en Java
- Utilisation des Optional pour éviter les nulls et améliorer la sécurité du code
- Programmation réactive avec Reactive Streams (frameworks comme Project Reactor)
- Combinaison de la POO et de la programmation fonctionnelle pour des architectures plus flexibles et lisibles

4. Architecture Java avancée

- Design Patterns : Application des patterns de conception pour résoudre des problèmes complexes (Singleton, Factory, Strategy, Observer, etc.)
- Création et gestion des microservices avec Spring Boot, communication entre services via RESTful APIs, gRPC, et WebSockets
- Gestion des transactions et des connexions avec JPA et Hibernate
- Introduction aux architectures hexagonales et clean architecture en Java

5. Programmation concurrente et multi-threading

- Compréhension des threads, ExecutorService, Callable et Future
- Synchronisation des threads : synchronized, Locks, Atomic variables
- Introduction aux CompletableFuture et aux concepts de la programmation asynchrone
- Résolution des problèmes de concurrence et des deadlocks dans un environnement Java multi-threadé

6. Tests unitaires et TDD (Test-Driven Development)

- Mise en œuvre des tests unitaires avancés avec JUnit 5 et Mockito
- Test-Driven Development (TDD) : Stratégies et bonnes pratiques pour la rédaction de tests en amont
- Tests d'intégration et tests fonctionnels
- Outils de couverture de code : JaCoCo, SonarQube
- Introduction à la gestion des tests dans des projets multi-modules

7. Utilisation des outils modernes et des bonnes pratiques de développement

- Mise en œuvre de CI/CD avec Jenkins, GitLab CI, ou GitHub Actions pour Java
- Gestion des dépendances avec Maven et Gradle
- Surveillance et gestion des logs avec SLF4J, Logback, et ELK Stack
- Sécurisation des applications Java : protection contre les attaques courantes (Injection SQL, XSS, CSRF)

8. Gestion de projets complexes et évolutifs

- Stratégies de développement pour des applications modulaires et évolutives
- Modularité avec le système de modules Java (JPMS - Java Platform Module System)
- Gestion des versions et des dépendances dans des environnements complexes
- Utilisation de Docker pour le développement et le déploiement d'applications Java en conteneurs

9. Cas pratiques et projets avancés

- Développement d'une application Java complexe intégrant des microservices, une base de données, et une architecture de communication (REST/gRPC)
- Application des concepts de programmation concurrente et de performances dans un projet réel
- Résolution de cas d'architecture Java complexes en groupe

JavaScript – Niveau Expert

Prix : 3030 € H.T.

Durée : 5 jours

Public cible : Développeurs ayant une maîtrise des bases de JavaScript et une expérience de développement web, souhaitant approfondir leur expertise dans les concepts avancés et les bonnes pratiques du langage JavaScript.

Prérequis : Maîtrise des concepts de base de JavaScript, du DOM, des fonctions et des événements, ainsi que des connaissances de base en HTML et CSS. Une expérience préalable en développement avec JavaScript est fortement recommandée.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir des architectures JavaScript complexes et maintenables
- Maîtriser les fonctionnalités avancées du langage (ES6+, async/await, modules, etc.)
- Optimiser les performances des applications JavaScript
- Utiliser des outils modernes de développement (Webpack, Babel, TypeScript)
- Appliquer des design patterns et des techniques avancées pour le développement d'applications web performantes et robustes

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Rappels et révision des bases avancées

- Fonctionnement de JavaScript : moteur V8, compilation et interprétation
- Les nouveautés ECMAScript 6 (ES6+) : let, const, classes, modules, template literals, destructuration
- Gestion des modules JavaScript avec import et export
- Promises, async/await pour la gestion des tâches asynchrones

2. Les fonctions avancées en JavaScript

- Fonctions de première classe et fonctions d'ordre supérieur
- Fonction bind, call et apply et leur utilisation dans des contextes spécifiques
- Manipulation avancée des closures et des portées (scope)
- Gestion des callbacks et inversion de contrôle (callback hell)
- Utilisation des générateurs (generators) et des itérateurs

3. Programmation fonctionnelle en JavaScript

- Concepts fondamentaux : immutabilité, pureté des fonctions, réutilisabilité
- Utilisation des méthodes de tableau avancées : map(), filter(), reduce(), some(), every()
- Fonctions d'ordre supérieur pour la composition de fonctions
- La notion de currying et des fonctions partielles

4. Gestion des erreurs avancée et debug

- Traitement des erreurs avec try/catch, gestion des erreurs asynchrones (promises)
- Meilleures pratiques pour la gestion des exceptions en JavaScript
- Techniques de débogage avancées (Utilisation de console, breakpoints dans les outils dev)
- Suivi des erreurs en production avec des outils comme Sentry

5. Architecture et design patterns en JavaScript

- Introduction aux design patterns : Singleton, Factory, Observer, Module, et MVC
- Utilisation des patterns pour l'architecture d'applications web complexes
- Gestion des états avec des architectures comme Flux et Redux

- Architecture modulaire avec JavaScript et gestion des dépendances avec ES Modules et Webpack

6. Optimisation des performances JavaScript

- Comprendre la boucle d'événements et le modèle de concurrence
- Gestion des memory leaks et optimisation de la consommation mémoire
- Profilage de performance : analyse du temps d'exécution avec des outils comme Chrome DevTools
- Optimisation des performances du DOM : techniques de batch updates, minimisation des reflows/repaints
- Meilleures pratiques pour le rendu côté client et la gestion des tâches asynchrones

7. Les API modernes en JavaScript

- Manipulation des Fetch API et des WebSockets pour les communications réseau
- Utilisation de l'API Fetch pour remplacer XMLHttpRequest
- Gestion des Service Workers et Web Workers pour la gestion de tâches en arrière-plan
- Introduction aux Progressive Web Apps (PWA)

8. TypeScript et JavaScript : Passage de JS à TS

- Introduction à TypeScript : pourquoi et comment migrer d'un projet JavaScript vers TypeScript
- Utilisation des types statiques et avantages de TypeScript dans la gestion des grandes bases de code
- Compilation et intégration avec des outils comme Webpack et Babel

9. Tests avancés en JavaScript

- Introduction aux tests unitaires avec Jest et Mocha
- Tests d'intégration, tests fonctionnels avec Cypress
- Mise en œuvre de la méthodologie TDD (Test-Driven Development) en JavaScript
- Mocking, spying et tests des fonctions asynchrones

10. Exercices pratiques et mini-projets avancés

- Développement d'une application complexe avec architecture modulaire, gestion d'état, et appels API
- Mise en œuvre de design patterns pour une gestion efficace des différentes parties de l'application
- Optimisation de la performance et des interactions utilisateur avec des techniques avancées de manipulation du DOM
- Intégration de tests unitaires, d'intégration et fonctionnels dans une application JavaScript complète

React – Débutant

Prix : 1900 € H.T.

Durée : 3 jours

Public cible : Développeurs débutants ou toute personne ayant une expérience en JavaScript et souhaitant apprendre à créer des applications web dynamiques avec React.

Prérequis : Maîtrise des bases de JavaScript (notamment les fonctions, objets, tableaux et événements). Une connaissance de base du HTML et du CSS est recommandée.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre les principes fondamentaux de React et son écosystème
- Créer des applications web réactives avec React
- Utiliser les composants, les états et les props pour construire des interfaces utilisateur dynamiques
- Gérer les événements utilisateur et manipuler le DOM via React
- Intégrer des API REST pour récupérer et afficher des données

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à React

- Qu'est-ce que React et son rôle dans le développement d'applications web modernes ?
- Les avantages de React : Réutilisation de composants, performance avec le Virtual DOM, unidirectional data flow
- Comprendre les composants : Class components vs. Functional components
- Installation de React via Create React App

2. Création d'un premier composant React

- Structure d'une application React : index.js, App.js
- Définition d'un composant avec des fonctions (stateless functional components)
- Rendu dynamique avec JSX (syntax sugar pour écrire du HTML dans JavaScript)
- Introduction aux props : Qu'est-ce que sont les props et comment les utiliser pour passer des données dans les composants

3. Les états dans React (useState)

- Introduction à l'état local dans un composant
- Gestion de l'état avec le hook useState
- Manipulation de l'état : changer l'état en fonction des actions de l'utilisateur (ex. : clic, saisie dans un champ)
- Interaction entre l'état et l'interface utilisateur : rafraîchissement du rendu lorsque l'état change

4. Gestion des événements dans React

- Introduction aux événements dans React (clics, changements, soumissions de formulaires)
- Utilisation des gestionnaires d'événements : onClick, onChange, onSubmit
- Passage des paramètres aux fonctions de gestion d'événements
- Liaisons entre la logique métier et l'interface utilisateur via les événements

5. Composants enfants et communication entre composants

- Passer des données entre composants via les props
- Création de composants réutilisables et dynamiques
- Prop drilling : Comment passer des données profondément dans la hiérarchie des composants
- Solutions pour la gestion de l'état global (introduction à Context API)

6. Introduction au Routing avec React Router

- Installation et configuration de React Router pour la gestion des pages
- Créer des liens de navigation avec Link et NavLink
- Gérer les routes et les paramètres de l'URL pour rendre des composants spécifiques
- Mise en place d'un rendu conditionnel pour afficher différentes vues selon l'URL

7. Travailler avec des API et gestion des données

- Introduction aux API REST : récupérer des données avec fetch()
- Utiliser useEffect pour effectuer des appels API lors du montage du composant
- Manipulation des données reçues : affichage dynamique des informations dans l'interface
- Gestion des erreurs et des chargements (ex : état de chargement, gestion des erreurs)

8. Gestion du formulaire dans React

- Création d'un formulaire simple avec gestion de l'état des champs
- Validation des entrées de formulaire : contrôle des erreurs avant soumission
- Utilisation de useState et onSubmit pour gérer la soumission des formulaires
- Implémentation de champs contrôlés et non contrôlés

9. Bonnes pratiques et organisation du code

- Structurer une application React : Organisation des dossiers et fichiers
- Modularité et réutilisabilité des composants
- Gestion des clés uniques avec key dans les listes de composants
- Introduction à la gestion des erreurs dans les applications React

10. Exercices pratiques et mini-projets

- Création d'une application de liste de tâches avec React (ajouter, supprimer, marquer comme complété)
- Développement d'une application météo avec appel à une API pour récupérer les données météorologiques et affichage des résultats
- Mise en place d'un formulaire interactif de contact avec validation et soumission des données

Angular – Niveau Expert

Prix : 2500 € H.T.

Durée : 4 jours

Public cible : Développeurs Angular confirmés souhaitant maîtriser les aspects avancés du framework pour concevoir, optimiser et maintenir des applications complexes en environnement professionnel.

Prérequis : Solide expérience d'Angular (niveaux débutant et intermédiaire maîtrisés), connaissance approfondie de TypeScript, des composants, services, formulaires réactifs, RxJS et du routing.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir des architectures évolutives et maintenables avec Angular
- Implémenter des performances optimisées et du lazy loading avancé
- Maîtriser RxJS pour des flux complexes
- Développer et tester des directives et composants réutilisables
- Gérer l'état de l'application avec NgRx
- Assurer la qualité, sécurité et testabilité du code Angular

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Architecture avancée d'une application Angular

- Organisation modulaire d'un projet à grande échelle
- Utilisation des FeatureModules, CoreModule, SharedModule
- Hiérarchisation et encapsulation des dépendances
- Définition d'une architecture scalable et testable

2. Optimisation des performances

- Change Detection : stratégie Default vs OnPush
- Utilisation avancée de trackBy, memoization avec pure pipes
- Chargement différé des modules (PreloadingStrategy, Quicklink)
- Optimisation des cycles de vie de composants (ngOnChanges, ngDoCheck)
- Détection et élimination des fuites de mémoire

3. RxJS avancé pour Angular

- Composition de flux avec des opérateurs complexes : concatMap, exhaustMap, combineLatest, withLatestFrom
- Gestion d'erreurs, annulation de requêtes, retry strategies
- Création de services orientés flux de données
- Application réactive : état, UI et appels API liés par RxJS

4. NgRx – Gestion d'état avec Redux

- Présentation du pattern Redux et du store NgRx
- Actions, reducers, selectors, effects : architecture complète
- Gestion des effets secondaires avec @ngrx/effects
- Optimisation des performances et lazy loading du store
- Alternative légère : ComponentStore pour des scopes plus localisés

5. Développement de composants avancés et librairies Angular

- Création de librairies Angular (ng generate library)
- Revue des mécanismes d'encapsulation CSS (ViewEncapsulation)

- Utilisation de ContentProjection et ngTemplateOutlet
- Création de composants dynamiques avec ComponentFactoryResolver
- Packaging et distribution d'une librairie interne

6. Sécurité et bonnes pratiques

- Prévention des vulnérabilités XSS, CSRF dans Angular
- Stratégies de sécurisation du routing et du backend
- Gestion des tokens et headers (JWT, OAuth) via HttpInterceptor
- Audit de dépendances et mises à jour (ng update, npm audit fix)

7. Tests avancés (unitaires et E2E)

- Bonnes pratiques de test avec Jest ou Karma/Jasmine
- Tests de composants avec dépendances : TestBed, mock services
- Tests d'effets avec @ngrx/effects/testing
- Tests E2E robustes avec Cypress : scénarios complexes et mocks

8. Déploiement et intégration continue

- Build production optimisé : ng build –configuration production
- Configuration d'environnement multi-cibles (environment.*.ts)
- Intégration continue : GitHub Actions, GitLab CI/CD
- Hébergement sur Firebase, Vercel ou Docker

9. Atelier final – Projet complet d'entreprise

- Création d'un tableau de bord sécurisé et modulaire
- Lazy loading, NgRx, routing imbriqué, composants partagés
- Déploiement simulé et revue de qualité du code

Vue.js – Débutant

Prix : 1900 € H.T.

Durée : 3 jours

Public cible : Développeurs front-end débutants ou confirmés souhaitant découvrir Vue.js pour créer des interfaces web modernes et interactives.

Prérequis : Bonnes bases en JavaScript, HTML et CSS. Aucune connaissance préalable de Vue.js requise.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre l'architecture et les principes de Vue.js 3
- Créer des composants interactifs et modulaires
- Gérer les données, événements, formulaires et directives
- Construire une application monopage simple avec Vue Router
- Organiser un projet Vue moderne avec Vite et Composition API

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à Vue.js

- Vue.js : historique, philosophie, place dans l'écosystème JS
- Avantages et cas d'usage
- Présentation des versions (Vue 2 vs Vue 3)
- Mise en place du projet avec Vite ou Vue CLI

2. Fondamentaux de Vue.js

- Vue Instance et cycle de vie
- Le DOM réactif
- Data binding : interpolation, v-bind, v-model
- Gestion des événements : v-on, raccourcis clavier, événements personnalisés

3. Composants Vue

- Définition et structure d'un composant
- Props et communication parent-enfant
- Slots (simples et nommés)
- Gestion de l'état local et réutilisabilité

4. Directives et templates

- Directives intégrées : v-if, v-for, v-show, v-bind, v-on
- Classes et styles dynamiques
- Utilisation des filtres (deprecated), remplacement par computed/methods
- Création de directives personnalisées simples

5. Composition API – Vue 3

- Introduction à la Composition API
- setup(), ref(), reactive(), computed(), watch()
- Comparaison avec Options API
- Organisation du code : composabilité, lisibilité, réutilisabilité

6. Gestion des formulaires

- Binding avec v-model
- Validation de base
- Introduction à des bibliothèques de validation comme VeeValidate

7. Navigation avec Vue Router

- Configuration du router
- Création de routes, redirections, routes dynamiques
- Navigation programmatique
- Passage de paramètres et vues imbriquées

8. Introduction à la gestion d'état (Pinia)

- Présentation de Pinia (successeur de Vuex)
- Création et utilisation d'un store simple
- Communication entre composants via store

9. Exercices pratiques et mini-projets

- Création d'une application TODO ou gestionnaire de notes
- Interface formulaire avec validation simple
- Mini dashboard avec plusieurs vues et composants

Node.js – Niveau intermédiaire

Prix : 1900 € H.T.

Durée : 3 jours

Public cible : Développeurs ayant déjà utilisé Node.js pour créer des scripts ou une API simple, souhaitant structurer des projets backend plus complexes.

Prérequis : Bonne maîtrise de JavaScript moderne et des bases de Node.js (Express, routage, gestion des requêtes et réponses, Promises).

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Structurer une application Node.js avec une architecture modulaire
- Implémenter des middlewares personnalisés et sécuriser les routes
- Intégrer et manipuler une base de données (MongoDB ou PostgreSQL)
- Gérer les erreurs de façon centralisée
- Documenter et tester leur API REST

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Architecture d'une application Node.js

- Structure MVC simplifiée (routes / contrôleurs / services)
- Modularisation du code, usage des modules ES
- Organisation des middlewares, gestion des dépendances

2. Express avancé

- Création de middlewares personnalisés (authentification, logs, erreurs)
- Route grouping, express.Router(), hiérarchie des routes
- Filtres et limitations d'accès (middleware conditionnel)
- Upload de fichiers avec multer

3. Gestion des erreurs et logs

- Centralisation des erreurs (middleware d'erreur)
- Formatage des erreurs personnalisées (AppError, errorHandler)
- Logging avec morgan, winston, logs structurés

4. Sécurité des API

- Authentification JWT (JSON Web Token)
- Hashage de mots de passe avec bcrypt
- Protection contre les injections, XSS, CSRF
- Limitation de débit (express-rate-limit), nettoyage des entrées (express-mongo-sanitize)

5. Base de données (MongoDB ou PostgreSQL)

- Connexion à une base MongoDB (avec mongoose) ou PostgreSQL (avec pg ou Prisma)
- Création de modèles (schemas, relations)
- Requêtes complexes, pagination, tri, filtres
- Gestion des migrations (pour SQL)

6. Appels asynchrones robustes

- Utilisation avancée de async/await

- Traitement de logique métier asynchrone
- Gérer les cas d'échec réseau ou base de données

7. Documentation de l'API

- Introduction à Swagger (via swagger-ui-express)
- Création d'un fichier de documentation OpenAPI
- Test interactif des endpoints via Swagger UI ou Postman

8. Tests et qualité

- Tests unitaires avec Jest (routes, services)
- Tests d'intégration API (supertest)
- Mock de base de données
- Intégration dans une CI (GitHub Actions en option)

9. Projet pratique

- Conception et développement d'une API complète :
- Authentification, CRUD complet, validation
- Base de données connectée
- Documentation Swagger
- Gestion des erreurs et sécurité

Next.js – Niveau Expert

Prix : 2300 € H.T.

Durée : 4 jours

Public cible : Développeurs expérimentés en React/Next.js souhaitant concevoir des applications web performantes, scalables et prêtes pour la production à grande échelle.

Prérequis : Bonne maîtrise de React, de JavaScript moderne, et de Next.js (routage, Server/Client Components, récupération de données).

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Structurer une application Next.js selon les meilleures pratiques (App Router, modularisation, performance)
- Maîtriser les Server Components, Server Actions, streaming et ISR avancé
- Mettre en œuvre des techniques d'optimisation du rendu, du chargement et du SEO
- Intégrer l'internationalisation, la gestion d'état et des middlewares avancés
- Déployer une application Next.js optimisée, sécurisée et maintenable

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Architecture avancée avec App Router

- App Router vs Pages Router : état de l'art
- Organisation d'un projet en modules, layouts imbriqués, providers globaux
- Optimisation du lazy loading et des bundles
- Gestion multi-domaines, micro-frontends (aperçu)

2. Server Components & Server Actions

- Approfondissement : composant purement serveur et client hybride
- Server Actions (Next.js 14) : appels directs à des fonctions côté serveur depuis des formulaires
- Sécurité des Server Actions, validation, revalidation automatique

3. Rendu dynamique et streaming

- Streaming partiel des pages (React Server Components)
- Utilisation de Suspense, loading.tsx et error.tsx
- Incremental Static Regeneration avancé (ISR dynamique, revalidateTag, revalidatePath)
- Détection automatique des chemins statiques et dynamiques

4. Internationalisation et routing avancé

- next-intl ou i18n natif de Next.js
- Routage conditionnel selon la langue
- Domaine dédié par locale
- Redirections et réécritures dynamiques (next.config.js)

5. API, middleware et Edge functions

- Création d'API REST et RPC via les routes route.ts
- Middleware Edge : gestion d'authentification, géolocalisation, A/B testing
- Middleware de rewriting conditionnel (edge config, headers dynamiques)
- Introduction à Vercel Edge Runtime

6. Sécurité et bonnes pratiques

- Authentification sécurisée (JWT, OAuth2, NextAuth.js)
- Server Actions et validation des inputs avec zod
- CSP, headers HTTP sécurisés (helmet, next-secure-headers)
- Sécurisation des endpoints API et rate limiting

7. Gestion d'état & intégration des outils

- Appel d'API avec SWR ou React Query
- Gestion d'état globale (Zustand, Redux Toolkit, ou Context API optimisé)
- Intégration de CMS headless : Sanity, Strapi, Contentful
- Intégration d'une base de données avec Prisma + PlanetScale ou PostgreSQL

8. Performances & accessibilité

- Audit Lighthouse, Web Vitals et optimisation des scores
- Optimisation des images, lazy loading et CDN
- Optimisation du temps de TTFB avec le rendu serveur
- Accessibilité : bonnes pratiques ARIA, navigation clavier, focus management

9. Déploiement et CI/CD

- Environnements de staging / production sur Vercel
- Configuration des variables sécurisées, preview URLs
- Pipeline GitHub Actions, checks automatiques (lint, tests, build)
- Monitoring avec Vercel Analytics, LogRocket ou Sentry

10. Projet final – Application professionnelle complète

- Développement d'une app complète avec :
- Authentification NextAuth.js
- Pages dynamiques avec Server Components + Server Actions
- CMS intégré (ou base de données Prisma)
- Middleware Edge pour sécurisation
- Déploiement et documentation

NestJS – Niveau Expert

Prix : 2300 € H.T.

Durée : 4 jours

Public cible : Développeurs back-end confirmés, lead dev ou architectes souhaitant concevoir des API robustes, modulaires et scalables avec NestJS.

Prérequis : Très bonne maîtrise de TypeScript, NestJS (modules, contrôleurs, services, DTO, validation), expérience avec base de données relationnelle ou NoSQL.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir une architecture modulaire évolutive (DDD, Clean Architecture, CQRS)
- Gérer des systèmes complexes avec les patterns de NestJS avancés
- Implémenter des microservices, de la communication inter-service (MQ, EventBus)
- Sécuriser les API au niveau middleware et infrastructure
- Assurer la qualité et la maintenabilité via tests, logging, CI/CD

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Architecture avancée

- Structure DDD et Clean Architecture avec NestJS
- Layers : controllers, use-cases, repositories, entities
- Injection dynamique, configuration multi-modules, dépendances croisées
- Feature-based vs domain-based folder structure

2. CQRS avec NestJS

- Introduction au pattern Command Query Responsibility Segregation
- Mise en place avec le module @nestjs/cqrs
- Gestion des handlers, buses, events
- Séparation stricte des lectures/écritures
- Exemple d'implémentation sur une API CRUD complexe

3. Microservices et communication interservices

- NestJS Microservices : architecture, transporteurs supportés (TCP, Redis, MQTT…)
- Communication via Events (event-driven) et Messages (RPC)
- Mise en place avec Redis / NATS
- Scalabilité, résilience, retry et timeout

4. Sécurité avancée

- Authentification OAuth2, JWT, gestion des rôles (RBAC)
- Guards customisés pour accès granulaire
- Protection CSRF, CORS avancé, Helmet
- Logs d'audit, surveillance de requêtes sensibles

5. Gestion de la configuration et des environnements

- Utilisation avancée de @nestjs/config
- Injection de configuration dynamique (multi-environnement, secrets)
- Séparation dev/staging/prod avec Docker ou CI/CD

6. Tests avancés

- Tests unitaires avec Jest : services, controllers, pipes, guards
- Tests d'intégration : setup in-memory DB (SQLite, Mongo Memory Server)
- Mocking avancé avec jest-mock, ts-mockito
- Tests E2E avec Supertest

7. Observabilité et logs

- Interceptors pour tracking et transformation des réponses
- Logging avancé avec @nestjs/logger, Winston ou Pino
- Intégration avec Elastic, Grafana, Prometheus (via exporters)

8. CI/CD et déploiement

- Pipeline CI avec GitHub Actions / GitLab CI : lint, test, build
- Dockerisation d'une app NestJS avec multi-stage build
- Déploiement sur Kubernetes, Fly.io, Render ou AWS (aperçu)
- Health check, readiness, monitoring d'app

9. Projet final – API hexagonale et scalable

- Création d'une API complexe (ex : gestion RH ou e-commerce) :
- Architecture modulaire + CQRS
- Authentification JWT
- Communication asynchrone via Redis ou NATS
- Tests et Dockerisation

C# – Débutant

Prix : 1950 € H.T.

Durée : 3 jours

Public cible : Développeurs débutants ou développeurs venant d'autres langages (Java, JS, Python) souhaitant apprendre C# pour développer des applications .NET.

Prérequis : Connaissances de base en algorithmique ou en programmation dans un autre langage.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre les fondements du langage C# et du framework .NET
- Développer des applications console et poser les bases pour le développement web ou desktop
- Maîtriser les types de données, structures de contrôle, collections et fonctions
- Utiliser la programmation orientée objet avec C#
- Lire, écrire et manipuler des fichiers et des données simples

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à C# et .NET

- Présentation de la plateforme .NET (Core, Standard, .NET 8)
- Environnement de développement : Visual Studio, Visual Studio Code
- Structure d'un projet C#
- Compilation, exécution, gestion des packages NuGet

2. Bases du langage

- Types de données : types primitifs, chaînes, dates, booléens
- Opérateurs, conversions, nullabilité
- Structures de contrôle : if, switch, while, for, foreach
- Fonctions, arguments nommés et optionnels
- Notions sur les tuples et pattern matching (C# 8+)

3. Programmation orientée objet

- Classes, objets, constructeurs, propriétés
- Méthodes et surcharge
- Encapsulation, héritage, polymorphisme
- Interfaces et classes abstraites
- Utilisation des records (C# 9+) et init-only properties

4. Collections et structures de données

- Tableaux, listes, dictionnaires
- Itération, recherche, tri
- Introduction aux LINQ (Language Integrated Query)
- Méthodes d'extension (extension methods)

5. Gestion des exceptions

- Try, catch, finally
- Création d'exceptions personnalisées
- Bonnes pratiques de gestion d'erreurs

6. Fichiers et manipulation des données

- Lecture/écriture de fichiers texte (File, StreamReader, StreamWriter)
- Sérialisation JSON (System.Text.Json)
- Manipulation de dates, formats et cultures

7. Introduction aux applications console et à l'interactivité

- Interactions avec l'utilisateur (Console.ReadLine / WriteLine)
- Gestion des arguments en ligne de commande
- Création d'outils CLI simples

8. Projet fil rouge

- Développement d'une mini-application de gestion (contacts, tâches, etc.)
- Interface console simple
- CRUD avec enregistrement en fichier JSON
- Application des principes objets et de la structuration du code

ASP.NET Core – Niveau Expert

Prix : 2900 € H.T.

Durée : 4 jours

Public cible : Développeurs back-end expérimentés, lead developers, architectes logiciels souhaitant concevoir des API ou des applications web complexes, sécurisées et performantes avec ASP.NET Core.

Prérequis : Très bonne maîtrise du C# (niveau avancé) et d'ASP.NET Core (contrôleurs, middleware, EF Core, routage, configuration).

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir des API web et des architectures modulaires robustes (Clean Architecture, DDD)
- Implémenter des stratégies avancées de sécurité, de performance et de mise en production
- Utiliser les dernières pratiques en matière de tests, de logs, de CI/CD et d'observabilité
- Gérer des scénarios complexes de persistance, mapping, validation, et monitoring

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Architecture logicielle avancée

- Clean Architecture, Domain-Driven Design (DDD) appliqué à ASP.NET Core
- Organisation modulaire : séparation entre Web, Application, Domain, Infrastructure
- Principes SOLID, découplage avec interfaces, injection de dépendances avancée
- Utilisation des MediatR, AutoMapper, FluentValidation

2. Création d'API REST avancées

- Conventions REST : statuts, routes imbriquées, HATEOAS (vue d'ensemble)
- Filtrage, pagination, tri dynamiques
- Versioning d'API (URL, headers, négociation de contenu)
- Contrôleurs génériques et conventions

3. Middleware et pipeline HTTP personnalisé

- Middleware custom : gestion centralisée des erreurs, journalisation, authentification
- Filtres (filters) et attributs personnalisés
- Utilisation fine des IApplicationBuilder, IEndpointRouteBuilder

4. Sécurité et authentification

- Authentification JWT et OAuth2 (Azure AD, IdentityServer, Auth0)
- Autorisation par rôles, policies, claims
- Protection CSRF, CORS avancé, rate limiting (AspNetCoreRateLimit)
- Sécurisation des données sensibles (KeyVault, chiffrement, Identity)

5. Entity Framework Core avancé

- Mapping complexe (relations, types possédés, value objects)
- Optimisation des requêtes LINQ et projections avec Select
- Transactions, verrouillages, stratégie de concurrence
- Multi-tenancy, audit, stratégie soft delete

6. Tests, logs et observabilité

- Tests unitaires / d'intégration avec xUnit, WebApplicationFactory, EF InMemory

- Mocks complexes avec Moq, FakeItEasy, Bogus
- Logging structuré avec Serilog et enrichissement (correlation ID, user context…)
- Traces et métriques avec OpenTelemetry, Grafana, Application Insights

7. Performances, résilience et mise en production

- Caching : mémoire, distribué (Redis), HTTP headers (ETag, Last-Modified)
- Retry, circuit breaker avec Polly
- Compression, minification, gestion fine du pipeline
- Publication sur Azure, Docker, Kubernetes ou Render

8. CI/CD et DevOps

- Pipelines GitHub Actions / GitLab CI : lint, tests, build, déploiement
- Dockerisation d'une API ASP.NET Core avec configuration multi-env
- Intégration avec Helm, Kustomize, Terraform (notions)
- Stratégies de migration de base et déploiement blue/green

9. Projet final

- Conception complète d'une API métier modulaire (ex : système RH, e-commerce, CRM)
- Architecture hexagonale avec MediatR
- JWT, logins sécurisés, validation centralisée
- Logs structurés, tests automatisés, déploiement Docker

Codage sécurisé en C/C++

Prix : 2100 € H.T.

Durée : 3 jours

Public cible : Développeurs C et/ou C++ intervenant sur des applications critiques (embarqué, système, temps réel, industriel, finance, cybersécurité), responsables qualité, architectes logiciels.

Prérequis : Maîtrise des fondamentaux du langage C ou C++ (pointeurs, structures, fonctions, gestion mémoire, compilation). Une expérience de développement concret est fortement recommandée.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Identifier les principales vulnérabilités dans le code C/C++
- Appliquer les bonnes pratiques de programmation défensive
- Rédiger un code robuste, fiable et sécurisé dès la conception
- Utiliser des outils d'analyse statique et dynamique pour prévenir les failles
- Se conformer aux référentiels de sécurité reconnus (CERT, MISRA, ISO/IEC TS 17961)

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à la sécurité logicielle

- Enjeux de sécurité dans le développement natif
- Historique et vulnérabilités emblématiques (Heartbleed, Stack Clash…)
- Panorama des normes : CERT C, CERT C++, MISRA C, ISO/IEC TS
- Attaques classiques : buffer overflow, use-after-free, integer overflow, format string

2. Failles courantes en C/C++

- Problèmes liés à la gestion mémoire manuelle :
- Déréférencement de pointeurs invalides
- Double free, fuites mémoire, free non initialisé
- Failles d'initialisation et d'accès concurrent
- Dangereuses fonctions standard (strcpy, sprintf, gets…)

3. Bonnes pratiques de codage sécurisé

- Validation stricte des entrées/sorties (whitelisting, bornes)
- Gestion défensive des pointeurs et allocations dynamiques
- Utilisation correcte des types (size_t, uint32_t, conversions sûres)
- Programmation robuste : assertions, invariants, defensive programming

4. Codage sécurisé en C++

- Exceptions vs codes d'erreur, RAII et gestion des ressources
- Bonnes pratiques avec les smart pointers (unique_ptr, shared_ptr)
- Conteneurs STL sécurisés vs accès mémoire direct
- Gestion de l'héritage, override explicite, règles d'encapsulation

5. Recommandations CERT C/C++

- Principes de la norme CERT (rules vs recommendations)
- Exemples de règles : INT32-C, EXP33-C, ARR30-C, MEM50-CPP
- Cas pratiques : analyse de code et refactoring à partir d'extraits non conformes

6. Outils d’analyse statique et dynamique

- Analyse statique : Clang-Tidy, Cppcheck, SonarQube, Coverity
- Analyse dynamique : Valgrind, AddressSanitizer, UndefinedBehaviorSanitizer
- Intégration dans les chaînes CI/CD (GitHub Actions, GitLab CI)
- Métriques de sécurité et rapports de conformité

7. Sécurisation du cycle de développement

- Intégration des contrôles sécurité dans le SDLC (Secure Development Life Cycle)
- Revue de code orientée sécurité
- Tests fuzzing (libFuzzer, AFL), injection de fautes
- Techniques de durcissement du binaire : ASLR, DEP, stack canaries, PIE

8. Études de cas et atelier final

- Analyse de failles réelles dans des bibliothèques open source
- Réécriture sécurisée d’un module critique (lecture/écriture fichier, socket, parsing)
- Application d’outils d’analyse et rédaction de rapports de non-conformité

Symfony – Niveau Débutant à Intermédiaire

Prix : 2600 € H.T.

Durée : 4 jours

Public cible : Développeurs PHP souhaitant apprendre à concevoir des applications web robustes et maintenables avec le framework Symfony.

Prérequis : Bonne maîtrise du langage PHP orienté objet. Expérience de développement d'applications web (formulaires, bases de données) et notions de Composer, HTTP et MVC

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre l'architecture d'un projet Symfony
- Créer une application web en s'appuyant sur les composants Symfony
- Gérer les routes, les contrôleurs, les entités, les formulaires et la base de données avec Doctrine
- Appliquer les bonnes pratiques de sécurité et de structuration de projet
- Utiliser les outils intégrés de débogage, tests et configuration

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à Symfony

- Présentation du framework Symfony et de l'écosystème PHP-FIG (PSR)
- Architecture MVC, composants réutilisables
- Installation avec Composer et Symfony CLI
- Structure d'un projet Symfony (src, config, templates, public, etc.)

2. Routing et contrôleurs

- Définir une route avec annotations, YAML ou PHP
- Création de contrôleurs (Controller, Response, Request)
- Génération d'URLs, redirections
- Passage de paramètres dans les routes

3. Twig – Le moteur de templates

- Syntaxe de base de Twig
- Héritage de templates et blocs
- Filtres, fonctions et extensions
- Affichage conditionnel et boucles

4. Entités et base de données avec Doctrine ORM

- Configuration de la base de données (.env, Doctrine config)
- Création d'entités et mapping via annotations ou attributs PHP 8
- Migrations, relations (OneToMany, ManyToMany)
- Requêtes via repository et QueryBuilder

5. Formulaires Symfony

- Création et affichage de formulaires avec FormBuilder
- Champs de formulaires (text, textarea, choice, checkbox, file, etc.)
- Validation (constraints) et messages d'erreurs
- Traitement des données et persistance

6. Services, injections et container

- Création de services personnalisés
- Injection automatique (autowiring) et configuration via YAML
- Notion de container de services
- Tags, événements, listeners

7. Sécurité

- Gestion de l'authentification : login, logout, firewall, encodage des mots de passe
- Sécurité des routes (access control, rôle utilisateur)
- Création d'un système d'inscription et de connexion avec SecurityBundle
- CSRF, validation et protection contre les attaques courantes

8. Console, debug et outils

- Commandes utiles (bin/console, debug:*)
- Utilisation du Profiler Symfony (Barre de debug)
- Dumping et logging avec Monolog

9. Tests fonctionnels et unitaires (initiation)

- Mise en place de tests avec PHPUnit
- Tests des contrôleurs et du comportement des routes
- Simulation de requêtes HTTP

10. Mini-projet final

- Mise en œuvre d'une application complète (ex : blog, gestion de tâches, catalogue produits)
- Mise en pratique du routage, des entités, des formulaires et de la sécurité

TypeScript – Prise en main et développement robuste en JavaScript typé

Prix : 1900 € H.T.

Durée : 3 jours

Public cible : Développeurs JavaScript souhaitant structurer et sécuriser leur code. Développeurs front-end ou back-end utilisant des frameworks modernes (React, Angular, Node.js…) Équipes techniques adoptant TypeScript pour l'industrialisation de leurs projets.

Prérequis : Maîtrise des fondamentaux JavaScript (ES6+ : classes, modules, fonctions fléchées, promesses). Connaissances de base en développement web (HTML/CSS facultatif)

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre les principes du typage statique appliqué à JavaScript
- Configurer et compiler un projet TypeScript efficacement
- Créer des types personnalisés et utiliser les types génériques
- Migrer progressivement un projet JavaScript vers TypeScript
- Intégrer TypeScript avec des bibliothèques/frameworks courants

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à TypeScript

- Pourquoi TypeScript ? Typage statique, IntelliSense, refactoring sécurisé
- Fonctionnement du compilateur tsc
- Configuration de projet (tsconfig.json)
- Compilation, surveillance (–watch), intégration avec NPM

2. Types de base

- Types primitifs : string, number, boolean, null, undefined, any, unknown
- Tableaux, tuples, objets typés
- Types union (|), intersection (&)
- Enumérations (enum)
- Typage de fonctions (arguments, retour, callbacks)

3. Typage avancé et structures complexes

- Interfaces et types (interface vs type)
- Types optionnels, readonly, et indexés
- Fonctions génériques (<T>)
- Types utilitaires (Partial, Pick, Omit, Record, etc.)
- Narrowing, assertion de types, contrôle de flux
- Type guards personnalisés

4. Classes et programmation orientée objet

- Classes, constructeurs, modificateurs d'accès (public, private, protected)
- Propriétés et méthodes statiques
- Héritage, interfaces d'implémentation
- Décorateurs (bases, pour Angular ou NestJS)

5. Modules et gestion de projet

- Modules ES vs CommonJS

- Import/export de types et interfaces
- Organisation modulaire d’un projet TypeScript
- Compilation vers différentes cibles (ES2022, ES5, etc.)

6. Interopérabilité avec JavaScript

- Typage de bibliothèques JS via DefinitelyTyped (@types)
- Utilisation de bibliothèques non typées
- Migration progressive d’un projet JavaScript
- Ajout de déclarations globales (fichiers .d.ts)

7. Outils et bonnes pratiques

- ESLint + TypeScript : règles de qualité
- Intégration avec VSCode : IntelliSense, debug, lint
- Build & CI : intégration avec Webpack/Vite, tests (Jest + TS), Git hooks
- Bonnes pratiques de typage, patterns recommandés et erreurs à éviter

COBOL Initiation

Prix : 2100 € H.T.

Durée : 3 jours

Public cible : Cette formation s'adresse aux informaticiens qui souhaitent acquérir une formation opérationnelle pour développer ou maintenir des applications en Cobol.

Prérequis : Connaissances de base en programmation.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Présenter les intérêts de Mainframe par rapports aux autres infrastructures
- Maîtriser la syntaxe globale du langage COBOL
- Clarifier la structure d'un programme Cobol
- Définir les ressources d'un programme Cobol en données dans un environnement Batch

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Présentation de l'interface ISPF

- Accès à la machine IBM
- Historique des mainframes
- Architecture générale des mainframes
- Terminologie IBM
- Présentation des choix du menu principal (Edit, View et SDSF)
- View : affichage du contenu de la librairie
- Edit : Création et mise à jour du contenu de la librairie
- SDSF: Spool Search and display Facility
- Commande de base de l'éditeur ISPF.
- Les commandes lignes de l'éditeur

2. Les commandes JCL

- La carte de commande JOB
- La structure des champs de JOB Statements
- La carte de commande EXEC
- La structure des champs de EXEC Statements

3. Présentation du langage Cobol

- Présentation de la structure du langage COBOL
- Description technique de COBOL
- Composition d'un programme COBOL
- Règle de codage sur une ligne COBOL
- Règle d'utilisation des mots clés et des chaînes de caractères
- Déclarations des variables

4. Opérations sur les données (MOVE, COMPUTE, ADD, etc...)

- Traitements des variables MOVE
- Opération de calcul ADD
- Opération de calcul SUBSTRUCT
- Opération de calcul MULTIPLY
- Opération de calcul DIVIDE

- Opération de calcul COMPUTE

5. Procédure de Compilation d'un programme

- Examen des phases de compilation
- Les Librairies de stockage

6. Traitement conditionnel

- Instruction IF .... THEN .... ELSE
- Instruction EVALUATE
- Instruction PERFORM
- Instruction GO ... TO

7. Gestion des Tables (OCCURS)

- Définition d'une Table
- Déclaration d'une Table
- Indice d'adressage d'une Table
- Indexation pour adressage des Tables
- Mise à jour des Index par l'instruction SET

CICS – Développement d'Applications Transactionnelles sur Mainframe

Prix : 2300 € H.T.

Durée : 3 jours

Public cible : Développeurs COBOL, PL/I ou Assembler amenés à développer ou maintenir des programmes CICS, Analystes techniques travaillant sur des applications transactionnelles, administrateurs systèmes ou responsables de production souhaitant mieux comprendre l'architecture CICS. Toute personne intervenant dans un environnement IBM z/OS avec des applications en ligne

Prérequis : Connaissance de la programmation COBOL ou PL/I. Notions de base sur le fonctionnement d'un mainframe et des fichiers VSAM. Expérience préalable avec JCL (souhaitable)

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Comprendre l'architecture CICS et le rôle du système transactionnel
- Développer des programmes CICS en COBOL ou PL/I
- Utiliser les commandes CICS pour la gestion des écrans, des fichiers, et des bases de données
- Gérer les communications entre programmes et utilisateurs via les terminaux
- Intégrer CICS avec des bases de données DB2 ou des programmes batch

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Introduction à CICS

- Qu'est-ce que CICS ? Rôle et position dans une architecture mainframe
- Différences entre traitement batch et transactionnel
- Structure d'un environnement CICS : régions, tables, programmes, terminaux
- Concepts de transaction, tâche (task), unité de travail

2. Architecture et composants CICS

- Définition des ressources : PPT (Program Processing Table), FCT (File Control Table), TCT (Terminal Control Table)
- Gestion des terminaux et sessions utilisateurs
- Cycle de vie d'une transaction CICS
- Notion de pseudo-conversation

3. Programmation CICS avec COBOL

- Structure d'un programme COBOL CICS
- Utilisation des commandes EXEC CICS (RECEIVE, SEND, LINK, XCTL, RETURN, etc.)
- Échange de données avec les utilisateurs via des écrans BMS (Basic Mapping Support)
- Gestion des commutateurs de programmes et des transitions

4. Gestion des fichiers et des bases de données

- Accès aux fichiers VSAM sous CICS avec les commandes FILE CONTROL
- Gestion de la concurrence, verrouillage, et intégrité des données
- Intégration avec DB2 sous CICS (commandes SQL dans un contexte transactionnel)
- Transactions multi-ressources (Commit, Rollback, Syncpoint)

5. Écrans et interface utilisateur

- Introduction à BMS (Basic Mapping Support)
- Définition et création de maps (MAPSET, MAP)
- Envoi et réception de données via écran 3270

- Gestion des erreurs d'écran, champs protégés, curseur, attributs dynamiques

6. Contrôle de flux et appels de programmes

- Appels de programmes (CALL, XCTL, LINK) et passage de paramètres
- Chaînage de transactions et pseudo-conversations
- Gestion des commutateurs d'écran et navigation utilisateur
- Programmation structurée avec des modules appelés et retour contrôlé

7. Gestion des erreurs et sécurité

- Gestion des codes de retour, gestion des exceptions CICS
- Utilisation de HANDLE CONDITION et IGNORE CONDITION
- Sécurité CICS : contrôle d'accès, autorisations, interactions avec RACF
- Journaux et diagnostics en cas d'erreur

8. Outils de développement et supervision

- Utilisation de CEDA pour la définition des ressources
- Utilisation de CEMT pour la gestion dynamique des ressources (activation, fermeture)
- Supervision des transactions en cours, consommation CPU, et files d'attente
- Bonnes pratiques de développement et de test en environnement CICS

Jakarta EE – Niveau Avancé : Développement d'Applications et Microservices d'Entreprise

Prix : 2500 € H.T.

Durée : 4 jours

Public cible : Développeurs Java expérimentés souhaitant concevoir des applications Jakarta EE modernes et modulaires. Ingénieurs logiciels impliqués dans des projets de migration vers des architectures microservices. Architectes logiciels, DevOps, chefs de projets techniques sur des plateformes Java EE. Équipes techniques souhaitant conteneuriser ou orchestrer leurs applications EE.

Prérequis : Solide expérience avec Java SE et les composants principaux de Jakarta EE (Servlets, EJB, JPA, REST). Bonne connaissance de l'architecture n-tiers et des modèles de conception Java. Expérience souhaitée avec Maven ou Gradle, Git, et un serveur d'application Jakarta EE.

Objectifs pédagogiques

À l'issue de la formation, les participants seront capables de :

- Concevoir des architectures modulaires à base de microservices avec Jakarta EE
- Utiliser Jakarta REST, CDI, JPA et JTA dans un contexte décentralisé
- Sécuriser et orchestrer les services avec JWT, OAuth2, OpenAPI, etc.
- Conteneuriser et déployer leurs applications Jakarta EE dans Docker et Kubernetes
- Intégrer Jakarta EE avec des systèmes de messaging, d'authentification et des outils de supervision

Méthodes et moyens pédagogiques : Travaux pratiques : formation alternant théorie et pratique.

Modalités d'évaluation

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation

Programme

1. Rappels essentiels Jakarta EE

- Architecture Jakarta EE et standardisation du cloud-native
- Application Monolithique vs Microservices
- Révision rapide des composants clés : EJB, JPA, RESTful, CDI, JTA

2. Conception de microservices avec Jakarta EE

- Découpage fonctionnel et granularité des services
- Patterns : API Gateway, Service Registry, Circuit Breaker (Hystrix-like)
- Communication REST/JSON entre services avec JAX-RS
- Appels interservices via HTTP et gestion de latence

3. CDI et architecture modulaire

- Injection de dépendances avancée avec Jakarta CDI
- Events, Interceptors, Producers
- Portabilité du code métier dans un environnement de services découplés
- Modularisation avec Jigsaw (Java 9+) ou via Maven multi-modules

4. Persistance et transactions distribuées

- Gestion de la persistance dans un environnement microservices
- Réplication vs base de données centralisée
- JPA dans des contextes décentralisés
- Transactions distribuées avec JTA ou alternatives (Sagas)

5. Sécurité et authentification

- Implémentation d'OAuth2 / JWT avec Jakarta Security
- Sécurisation des API REST avec JAX-RS et filtres personnalisés
- Mise en œuvre de rôles, autorisations, et contexte utilisateur

- OpenID Connect (vue d’ensemble)

6. Documentation, supervision, résilience

- Génération de documentation avec OpenAPI / Swagger
- Health checks avec MicroProfile Health
- Configuration centralisée avec MicroProfile Config
- Metrics, tracing, et log centralisé avec Prometheus, Grafana, ELK

7. Conteneurisation et orchestration

- Création d’images Docker pour Jakarta EE (Payara, WildFly, TomEE)
- Déploiement d’un cluster de services sur Kubernetes ou OpenShift
- Gestion de la configuration via secrets, volumes, config maps
- Bonnes pratiques CI/CD avec Jenkins, GitHub Actions ou GitLab

8. Interopérabilité et migration

- Intégration avec des services Spring Boot, Quarkus, Node.js via REST ou Kafka
- Migration progressive d’un monolithe Java EE vers une architecture microservices
- Séparation des responsabilités et évolution continue
- Utilisation de Jakarta EE Core Profile et Lite Profile (Jakarta EE 10+)

Le Sur-Mesure

Avec notre équipe pédagogique, nos formateurs experts et nos consultants dédiés, nous co-construisons des formations personnalisées pour développer l'engagement et la performance de vos collaborateurs.

La conception de formations sur mesure

Votre consultant Skillia organise un entretien de cadrage avec l'expert-formateur pressenti afin de définir précisément les objectifs de la formation et d'évaluer les besoins de préparation. Spécialiste du domaine concerné, l'expert analyse les spécificités du projet, le profil des participants et élabore, avec vous, un cahier des charges détaillé garantissant la pertinence du dispositif.

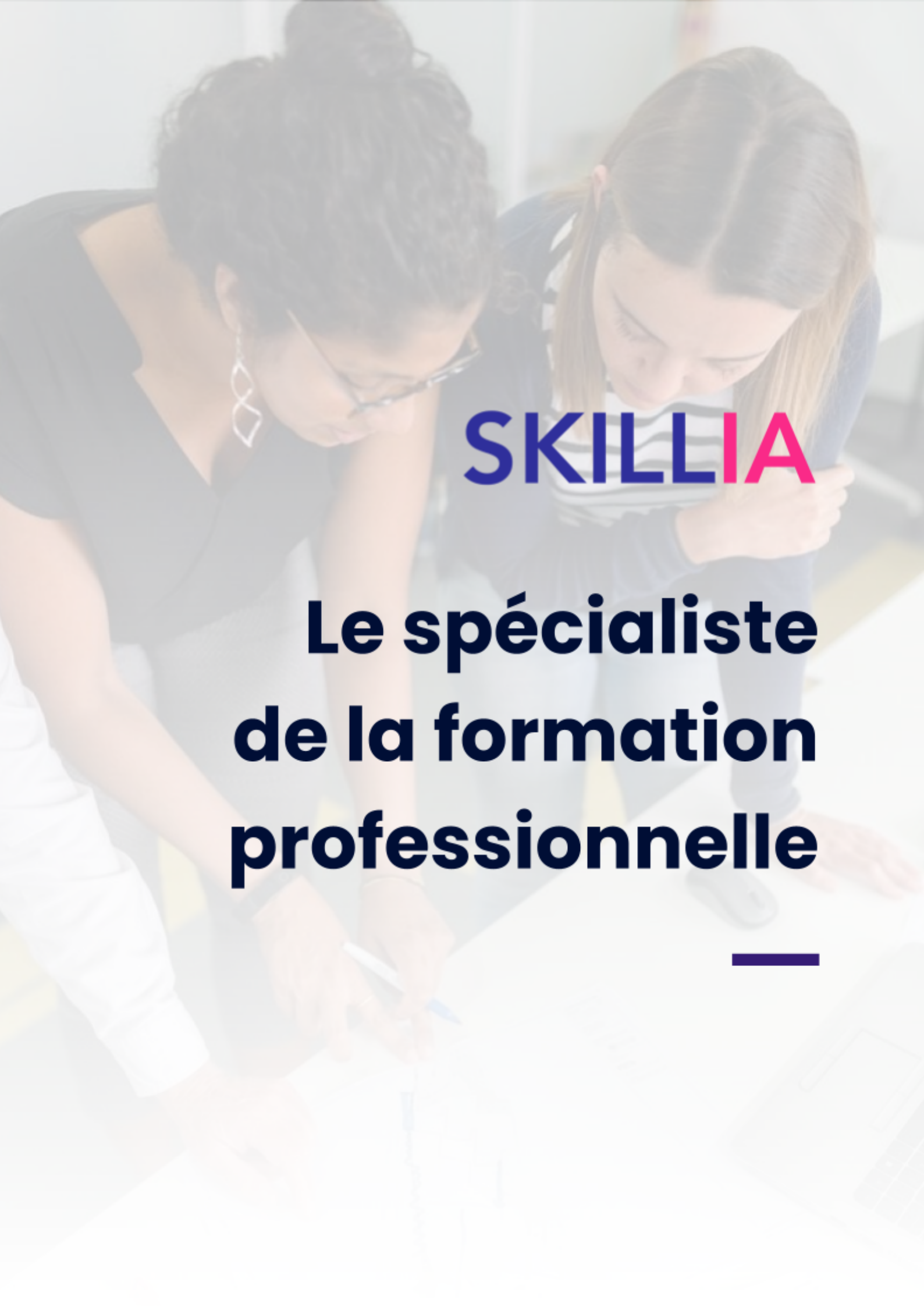
Des solutions e-learning personnalisées

Vous souhaitez une formation 100 % sur mesure, composée de modules e-learning et de vidéos adaptées à votre métier ?

Skillia met à votre disposition son savoir-faire et ses outils de production pour concevoir des formations digitales performantes, alliant interactivité, efficacité et accompagnement personnalisé par des tuteurs experts.

La gestion de projets de grande envergure

Qu'il s'agisse de former plusieurs équipes, sur différents sites ou dans plusieurs langues (français, anglais…), Skillia dispose de l'expertise nécessaire pour piloter l'ensemble du projet : audit, conception de programmes, construction de parcours, ainsi que coordination technique et logistique.



SKILLIA

Le spécialiste de la formation professionnelle