Blue AcademIA
Blue AcademIA est une marque de BlueMarketing
contact@blueacademia.fr
www.blueacademia.fr

Programme de formation : IA pour les développeurs : Coder plus vite et mieux avec l'IA générative

Une formation 100 % hands-on pour transformer votre façon de coder avec l'IA générative : assistants de code (GitHub Copilot, Cursor, Claude Code), génération et refactoring, tests et debugging assistés, revue de code, documentation et intégration d'un LLM dans vos applications, sans sacrifier la qualité ni la sécurité.

Durée : 21.00 heures ou jour(s) si formation en présentiel.

Lieu : À distance ou en présentiel

Profils des stagiaires

- Développeurs back-end, front-end et full-stack
- Ingénieurs logiciels
- Lead devs
- Intégrateurs
- Ingénieurs QA/test
- DevOps
- Développeurs en reconversion souhaitant adopter l'IA au quotidien

Prérequis

- Savoir programmer dans au moins un langage (Python, JavaScript/TypeScript, Java, C#…).
- Pratique courante de Git et d'une IDE moderne (VS Code, JetBrains).
- Aucune expérience préalable des assistants IA n'est requise.
- Les pré-requis sont validés lors d'un échange avec le commanditaire du projet de formation et retranscrits dans le document de recueil des besoins. Il s'agit d'un document collaboratif partagé systématiquement avec le commanditaire à l'issue du premier entretien pour co-construire l'action de formation. Par ailleurs, nous adressons à chaque apprenant, 2 semaines avant le début de la formation, un questionnaire de préparation de la formation pour adapter au mieux la formation à ses besoins.

Objectifs pédagogiques

- Comprendre concrètement comment les LLM « comprennent » et génèrent du code en 2026.
- Maîtriser les principaux assistants de code (Copilot, Cursor, Claude Code) et savoir quand utiliser chacun.
- Adopter le prompt engineering appliqué au code pour générer du code juste, lisible et testé.
- Accélérer la compréhension, le refactoring et la migration de code legacy.
- Industrialiser les tests, le debugging, la revue de code et la documentation avec l'IA.
- Intégrer proprement un LLM dans ses propres applications (API, SDK, structured output) et sécuriser ses usages.

Contenu de la formation

- Comment l'IA code en 2026 : ce qu'un développeur doit comprendre
  - Comment un LLM génère du code : tokens, contexte, complétion vs raisonnement.
  - Panorama 2026 des modèles de code : Claude Opus 4 / Sonnet 4.5, GPT-5, Gemini 2.5, modèles open-source (Qwen Coder, Codestral).
  - Forces et limites réelles : hallucinations d'API, dépendances obsolètes, faux sentiment de sécurité.
  - Panorama des outils : complétion inline, chat, mode agent, IDE intégrées vs CLI.
  - Démo comparée : faire résoudre une même tâche par 3 assistants et analyser les différences de qualité.
  - Atelier : repérer 5 erreurs subtiles dans du code généré par IA (API inventée, edge case oublié, faille).
  - Cartographier vos tâches de dev hebdomadaires et estimer le gain de temps réaliste par l'IA.
- Maîtriser son assistant de code : Copilot, Cursor, Claude Code
  - GitHub Copilot : complétion, Copilot Chat, edits multi-fichiers, mode agent.
  - Cursor & Windsurf : édition agentique, indexation du codebase, règles de projet (.cursorrules).
  - Claude Code & Cline : agents en CLI, exécution de commandes, modifications multi-fichiers pilotées.
  - Choisir le bon outil selon la tâche : complétion, refactoring large, exploration, automatisation.
  - Configurer Copilot et Cursor sur un vrai dépôt et écrire ses premières règles de projet.
  - Atelier : implémenter une nouvelle feature de bout en bout avec un assistant en mode agent.
  - Comparatif live : même refactoring multi-fichiers réalisé avec Copilot vs Cursor vs Claude Code.
- Prompt engineering appliqué au code
  - Anatomie d'un bon prompt de dev : contexte du projet, contraintes, conventions, format attendu.
  - Donner le bon contexte : fichiers pertinents, types, exemples de code existant, tests.
  - Techniques : few-shot, raisonnement étape par étape, demande de plan avant code.
  - Fichiers d'instructions persistants (CLAUDE.md, .cursorrules, copilot-instructions) pour cadrer l'assistant.
  - Atelier : transformer un prompt vague en prompt précis et mesurer l'écart de qualité du code produit.
  - Rédiger le fichier d'instructions projet (conventions, stack, do/don't) de votre équipe.
  - Cas pratique : générer une fonction complexe respectant vos conventions internes du premier coup.
- Génération, refactoring et code legacy
  - Génération guidée : du squelette de module à l'implémentation complète et typée.
  - Refactoring assisté : extraction, renommage, simplification, application de design patterns.
  - Comprendre du code legacy : expliquer, cartographier, documenter une base inconnue.
  - Migrations : changement de framework, de version de langage, conversion entre langages.
  - Atelier : comprendre et documenter un module legacy non commenté en moins de 30 minutes.
  - Refactorer une fonction de 200 lignes en composants testables, sous supervision.
  - Cas réel : migrer un script Python 2 / ancienne lib vers une version moderne et typée.
- Tests, debugging et qualité assistés par l'IA
  - Génération de tests unitaires et d'intégration (pytest, Jest) : couverture, cas limites, mocks.
  - TDD assisté par IA : écrire les tests d'abord, puis l'implémentation.
  - Debugging : analyse de stack trace, hypothèses de cause, reproduction d'un bug.
  - Qualité : détection de code smells, complexité, intégration avec SonarQube et linters.
  - Atelier : générer une suite de tests atteignant 80 % de couverture sur un module existant.
  - Cas pratique : diagnostiquer et corriger un bug réel à partir d'une stack trace.
  - Mettre en place une boucle « écrire test → générer code → vérifier » sur une feature.
- Revue de code, documentation et collaboration
  - Revue de code assistée : détection de bugs, de failles et de régressions sur une pull request.
  - Génération de documentation : docstrings, README, ADR, documentation d'API (OpenAPI).
  - Messages de commit et descriptions de PR clairs et normalisés.
  - Intégration dans le workflow Git/GitHub : Actions, revue automatisée, garde-fous.
  - Atelier : faire revoir une vraie pull request par l'IA et trier le vrai du faux positif.
  - Générer la documentation d'API (OpenAPI) et le README d'un service existant.

    - Mettre en place une revue de code IA dans une GitHub Action sur un dépôt de démonstration.
- Intégrer un LLM dans ses applications (API & SDK)
    - Appels API Anthropic / OpenAI : authentification, messages, paramètres, coûts.
    - Structured output et tool calling : extraire des données fiables et typées.
    - Prompt caching, streaming et gestion des erreurs / limites de débit.
    - Bonnes pratiques d'intégration : abstraction du fournisseur, tests, observabilité minimale.
    - Atelier : ajouter une fonctionnalité de classification de tickets à une petite application.
    - Implémenter une extraction de données structurées (JSON validé) à partir de texte libre.
    - Activer le prompt caching sur un cas d'usage répétitif et mesurer l'économie réalisée.
- Sécurité, conformité et bonnes pratiques durables
    - Risques de sécurité : code vulnérable généré, secrets exposés, dépendances douteuses.
    - Propriété intellectuelle et licences du code généré ; politiques entreprise (Copilot Business, zero-retention).
    - Confidentialité : ne pas exposer code propriétaire / données ; options self-hosted et souveraines.
    - Charte d'équipe et montée en compétences : où l'IA aide, où la relecture humaine reste obligatoire.
    - Atelier : auditer du code généré par IA à la recherche de failles et de secrets en dur.
    - Rédiger en équipe la charte d'usage de l'IA de développement (do / don't / relecture obligatoire).
    - Cas pratique : configurer un assistant en mode confidentiel (zero-retention) sur un dépôt sensible.

Organisation de la formation

Equipe pédagogique

Data Scientist, développeur et formateur expert en IA, Daouda DIOP accompagne les équipes techniques dans l'adoption des nouveaux usages du développement logiciel assisté par l'intelligence artificielle. Docteur en sciences, il utilise au quotidien GitHub Copilot, Cursor, Claude Code et les LLM pour concevoir des applications, automatiser des tâches de développement et améliorer la qualité du code. Son approche pédagogique, fondée sur la pratique et le cas de l'entreprise, permet aux participants d'intégrer rapidement ces outils dans leurs projets.

Moyens pédagogiques et techniques

- Dépôt GitHub de TP fourni, environnement VS Code prêt à l'emploi, accès aux assistants pendant la session.
- Modèle de fichier d'instructions projet et de charte IA de développement offerts.
- Supports pédagogiques commentés et partagés à distance
- Apports méthodologiques structurés
- Missions d'application intersession sur le cas réel de l'apprenant
- Cas pratiques
- Mise à disposition en ligne de documents supports à la suite de la formation
- Pour les formations en présentiel, des salles de formation agréables, lumineuses, spacieuses, modernes, équipées de connexion très haut débit, de grands écrans, et à proximité immédiate des transports.
- Pré-requis matériels : disposer d'un ordinateur. En cas de formation à distance, disposer également d'une connexion internet.
- Une semaine avant le début de la formation, nous donnons, à chaque participant, un accès sécurisé à notre plate-forme de formation en ligne (LMS : Learning Management System). L'apprenant accède via son extranet au livret d'accueil, au tuto de l'espace apprenant et au règlement intérieur.

Dispositif de suivi de l'exécution de l'évaluation des résultats de la formation

- Mini-défis de code par module + projet fil rouge réalisé avec l'IA, présenté en fin de formation.
- Attestation BlueMarketing Academy + badge « Développeur augmenté par l'IA » partageable.
- Suivi des missions d'application réalisées entre les séances
- Feuilles de présence

- Formulaires d'évaluation de la formation
- Mises en situation
- Certificat de réalisation de l'action de formation
- 10 jours avant l'entrée en formation : Nous transmettons à chaque stagiaire un questionnaire de préparation de la formation permettant d'adapter au mieux la formation à ses besoins.
- Pendant la formation : Le contrôle de connaissances et des acquis se font de façon continue via des quiz en ligne avec notation instantanée et des cas pratiques à réaliser. En fin de formation, les stagiaires restituent leurs travaux avec feedback du formateur.
- A l'issue de la formation : Nous adressons aux stagiaires un questionnaire de satisfaction à chaud et d'auto-diagnostic. Nous mesurons ainsi leur satisfaction et constatons directement l'évolution des compétences en comparaison avec les données recueillies avant la formation. Nous adressons un questionnaire d'évaluation au formateur.
- 15 jours après la fin de la formation : Nous adressons un questionnaire d'évaluation au financeur (OPCO).
- Un mois après la formation : Nous adressons un questionnaire d'évaluation aux différentes parties prenantes : commanditaire /manager.
- 3 mois après la formation : Nous transmettons au stagiaire un questionnaire d'évaluation à froid pour évaluer l'impact de la formation dans son activité professionnelle.

Tarif de la formation

À partir de 6300.00 €HT

TVA non applicable, art. 261-4-4 du CGI