WealCome
Expertise en conception logicielle
BDD / TDD / DDD / Clean Architecture

PROGRAMME DE FORMATION

Document généré le 23/11/2026

Agents IA et Software Craft : la vitesse sans la dette — Monde React / TypeScript

Type d'action : **Action de formation** Durée totale : **14h**

Informations de session

Lieu : Visio par Zoom

Formateur : Michaël AZERHAD - Président de S.A.S.U WealCome et architecte / expert technique depuis plus de 20 ans

Dates et horaires

Date	Début	Fin	Durée
21/12/2026	09:15	12:15	3h
21/12/2026	13:00	17:00	4h
22/12/2026	09:15	12:15	3h
22/12/2026	13:00	17:00	4h

Description

Tout le monde produit vite avec l'IA. Peu de gens produisent vite, et longtemps.

Sur un frontend React, l'IA livre en quelques minutes des écrans entiers, des hooks et des stores. Sans Software Craft, ce qu'elle produit ne tient pas la distance :

- une logique métier noyée dans les composants, le JSX et les useEffect en cascade ;
- un état éparpillé entre Context, hooks et cache serveur, que plus personne ne maîtrise ;
- des tests écrits après coup, qui rassurent sans réellement protéger ;
- du code en trop, des hallucinations, des fonctionnalités que personne n'a demandées ;
- un contexte qui sature et des tokens qui explosent en allers-retours ;
- du code relu ligne à ligne, sans savoir vraiment le juger ;
- un projet qui bascule en legacy en quelques mois au lieu de quelques années.

Cette formation prend le problème à la racine.

En deux jours, vous menez un vrai projet de bout en bout : une spécification brute, mal ficelée, telle qu'un fondateur vous la donnerait, jusqu'à un front React modulaire où plusieurs features tournent ensemble.

Ce sont les agents IA qui produisent ; c'est vous qui décidez.

Technologies :

React 19+ / Vite

TypeScript, version courante

Context React (branche sans state manager)

Redux RTK + RTK Query

Zustand + TanStack Query

Vitest / Testing Library

Stryker (mutation testing)

Purify-ts, Effect-TS (gestion d'erreurs typée)

Claude Code : agents et skills conçus par Michaël AZERHAD

Objectifs de la formation

À l'issue de la formation, les participants auront acquis les connaissances et méthodes nécessaires pour :

- Comprendre les principes du Software Craft : agilité, NoEstimates, DDD, BDD, TDD, refactoring et Clean / Hexagonal Architecture.
- Maîtriser les stratégies de tests automatisés (unitaires, intégration, end-to-end, mutation testing) et les principes de qualité logicielle.
- Comprendre les approches de conception métier et les spécificités du DDD appliqué au frontend React.
- Maîtriser les principes d'une architecture frontend modulaire, testable et évolutive, fondée sur les use cases, Domain Models, View Models et adaptateurs.
- Comprendre les stratégies de gestion d'état avec Redux RTK, Zustand, TanStack Query et React Context, ainsi que leurs compromis.
- Appréhender la programmation événementielle, les flux asynchrones, les optimistic updates, la gestion des erreurs et l'optimisation des rendus.
- Comprendre les principes d'orchestration des agents IA, de la spécification BDD à la livraison d'une application fonctionnelle.
- Maîtriser les méthodes de supervision et de contrôle du code généré par l'IA, en limitant hallucinations, sur-implémentation et dette technique.
- Développer une approche critique des choix architecturaux et savoir défendre les pratiques Software Craft au sein d'une équipe.

Prérequis

- Solide expérience du développement (au moins 5 ans recommandés)
- Bonne maîtrise de TypeScript et de React
- Notions de state management appréciées (Redux, Zustand ou autre)
- Bonnes connaissances en programmation orientée objet et/ou fonctionnelle
- Capacité à écrire un simple test unitaire avec Jest ou Vitest
- Aucune expérience préalable des agents IA requise. « Niveau expert » désigne le niveau de pratique visé, pas un prérequis.

Public visé

- Technical Leader
- Développeur Frontend
- Développeur Full Stack
- Architecte technique

Méthodes pédagogiques

- Live coding et orchestration d'agents en direct par le formateur
- Mob programming : les participants codent et orchestrent ensemble
- Apports théoriques émergeant de la pratique, sur un cas métier réaliste
- Comparaisons entre code écrit à la main et code produit par les agents, et entre les différentes approches de state management
- Challenges proposés sur le TDD, la Clean Architecture, le state management et le refactoring

Les agents Claude Code du formateur sont utilisés tout au long des deux jours, et non cantonnés à une démonstration.

Selon le curseur IA choisi par le groupe, les participants observent, puis orchestrent eux-mêmes les agents : spécification, tests, code, validation par les tests et refactoring.

Capacités développées

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

Axe 1 — Software Craft

- Identifier le « produit dans le produit » et découper les besoins en itérations courtes selon NoEstimates.
- Appliquer les principes du DDD au frontend : langage ubiquitaire, parcours entre Bounded Contexts et délimitation des responsabilités front / back.
- Rédiger des spécifications BDD non ambiguës, centrées sur les comportements et exploitables comme tests d'acceptation.
- Pratiquer le TDD Outside-In sur React, en partant des composants ou des use cases.
- Concevoir des tests unitaires, d'intégration et end-to-end in-memory avec Vitest et Testing Library ; distinguer tests sociables / solitaires et utiliser les doublures adaptées.
- Appliquer le mutation testing avec Stryker, comprendre son lien avec le TDD et viser 100 % de couverture des mutations pertinentes.
- Refactorer en continu le code de production et les tests sans régression.
- Concevoir un Domain Model frontend, identifier ses défauts et choisir entre un modèle global ou par use case.
- Structurer une application React en Clean / Hexagonal Architecture : use cases, Domain Models, View Models pilotés par les tests (selectors Redux, presenters Zustand ou sans state manager), états d'écran et composants « humble objects ».
- Concevoir et interchanger les adaptateurs secondaires : backend en mémoire, stockage navigateur et API tierces.
- Comparer les stratégies de gestion d'état (Redux RTK, Zustand avec TanStack Query, React Context), choisir la plus adaptée et positionner les use cases.
- Mettre en œuvre une architecture événementielle avec Redux et la comparer aux appels directs d'un store Zustand.
- Distinguer état serveur et état client, maîtriser les re-renders et optimiser les performances.
- Implémenter en TDD des optimistic updates et des flux asynchrones.
- Organiser un frontend modulaire avec un routing piloté par les use cases, indépendamment du découpage métier du backend.
- Gérer les erreurs avec des types Either ou des exceptions, en évaluant leurs compromis jusqu'à l'affichage des états d'erreur.

Axe 2 — Développement agentique

- Orchestrer des agents IA de la spécification à l'application exécutable, avec supervision des décisions ou délégation en mode « trust ».
- Produire des spécifications BDD structurées, catégorisées, séquencées et suivies par statut de complétion.
- Superviser le TDD réalisé par les agents et prévenir hallucinations, code superflu et sur-implémentation.
- Contrôler la qualité du code généré sans relecture systématique ligne à ligne, grâce aux spécifications, tests, mutation testing et règles architecturales.
- Évaluer, corriger et justifier les choix de conception proposés par l'IA selon les principes du Software Craft.
- Configurer les agents, commandes « décorateurs » et hooks pour fiabiliser les workflows.
- Faire générer les environnements de développement, configurations et jeux de données par story.

- Selon les options retenues : refactorer avec les agents un frontend legacy généré par IA vers une architecture testable et maintenable.

Capacités transversales

- Argumenter et promouvoir les pratiques Software Craft auprès d'une équipe, y compris auprès de profils sceptiques.
- Concilier productivité apportée par l'IA, qualité du code, maintenabilité et maîtrise des décisions techniques.

Modalités d'évaluation

- QCM en fin de formation
- Mise en pratique : les participants pilotent eux-mêmes les agents sur le projet fil rouge
- Échanges et discussions

Documents et supports pour les apprenants

- Supports de cours en markdown très complets
- Les agents et skills Claude Code du formateur (orchestrateur, TDD, spécification BDD et bien d'autres), rodés sur plus de 200 000 lignes de code en production
- Projet Git du live coding avec toutes ses branches : sans state manager, et une par state manager
- Partage de ressources (articles, blogs, discussions de forums tech)
- Discussions privées sur le canal Slack dédié au mini-groupe post-formation

Déroulé de la formation

Jour 1 — Spécifier et livrer des features React multi-scénarios

Matin — Du besoin brut à la première feature en TDD

- Accueil des participants, tour de table, présentation des parcours et recueil des attentes.
- Présentation du projet fil rouge : frontend React complexe, interface fournie et backend simulé en mémoire pour se concentrer sur les comportements.
- Quiz oral de positionnement sur les fondamentaux du Software Craft.
- Fondamentaux de l'agilité : approche produit, itérations courtes et NoEstimates.
- Analyse d'une spécification métier brute, remise en question des pratiques courantes et identification des besoins réels.
- Identification du « produit dans le produit » avec des agents agilistes : exploration d'au moins cinq orientations business, priorisation et choix humain du périmètre.
- DDD appliqué au frontend : langage ubiquitaire, parcours entre Bounded Contexts, responsabilités front / back et modularisation indépendante du découpage backend.
- Atelier BDD avec agent IA : clarification par QCM, scénarios centrés sur les comportements et spécifications catégorisées, séquencées, numérotées et suivies par statut.
- Articulation entre macro-architecture, découpage en features et TDD.
- Clean / Hexagonal Architecture frontend : use cases, Domain Model, View Models, adaptateurs, composants React « humble objects » et place des state managers au cœur de l'architecture.
- Stratégies de tests : tests unitaires et d'intégration avec Vitest, tests end-to-end in-memory avec Testing Library.
- Démarrage de la première feature en ATDD / TDD Outside-In avec agents IA : approche composants ou use cases, cycle red – clean green but minimal – maybe refactor further, tests sociables et programming by wishful thinking.
- Supervision des agents TDD spécialisés pour limiter les hallucinations, le code superflu et la sur-implémentation.
- Mutation testing avec Stryker : lien avec le TDD, objectif de 100 % et maîtrise de la qualité du code généré.

Après-midi — De la première feature opérationnelle à plusieurs features

- Implémentation progressive des règles métier, use case par use case, avec agents IA, refactoring continu et maintien d'un mutation testing à 100 %.
- Émergence du Domain Model à partir des tests : modèle global ou par use case, erreurs courantes, refactoring et validation humaine des choix de l'IA.
- Comparaison des approches de gestion d'état : React Context, Redux RTK / RTK Query et Zustand / TanStack Query ; choix collectif de la branche développée en direct, les autres étant finalisées par le formateur pour le lendemain.
- Conception des View Models pilotée par les tests : selectors Redux, presenters Zustand ou sans state manager, états de chargement, d'erreur et d'absence de données.
- Adaptateurs secondaires : backend en mémoire simulant latence et erreurs, stockage navigateur (localStorage, IndexedDB) et API tierces, avec tests d'intégration.
- Intégration des composants React comme adaptateurs primaires.

- Orchestration multi-agents : décomposition et coordination des tâches, supervision des décisions, délégation en mode « trust » et démonstration du pilotage à distance de Claude Code.
- Mise en service de la première feature avec Vite : environnement de développement, données de test par story, configuration automatisée et substitution des adaptateurs selon l'environnement.
- Développement de nouvelles features proposées par les participants : ateliers BDD, implémentation agentique, évolution maîtrisée de l'architecture et maintien du mutation testing à 100 %.
- Exercices en partage d'écran, questions-réponses et remise des supports pédagogiques illustrés à partir du code réalisé.

Jour 2 — Passer à l'échelle avec un frontend modulaire

Matin — Conception avancée

- Tour de table, retour sur les acquis du jour 1 et clarification des notions.
- Comparaison des implémentations avec et sans state manager : Redux RTK, Zustand et React Context, positionnement des use cases et compromis architecturaux.
- Optimisation des re-renders : selectors, mémoïsation et performances.
- Programmation événementielle Redux : actions, découplage des modules et comparaison avec les appels directs d'un store Zustand.
- Testabilité des use cases Redux : tests sociables sur store réel sans mocks ni wrappers, Given / Then, injection des gateways via extraArgument et fakes sans ports artificiels.
- Routing et navigation pilotés par les use cases, testables indépendamment des composants.
- Approfondissement du Domain Model : exercices de modélisation avec et sans IA, analyse critique et correction des choix proposés par les agents.
- Distinction état serveur / état client : RTK Query, TanStack Query et Zustand, selon les responsabilités de chaque solution.
- Optimistic updates en TDD : mises à jour anticipées, gestion des échecs et rollback avec backend simulé.
- Orchestration asynchrone : enchaînement des use cases, listener middleware Redux et logique événementielle inspirée des Process Managers.
- Modularisation frontend : émergence de nouveaux modules, communication entre Bounded Contexts et découplage par événements sans reproduire l'architecture backend.

Après-midi — Maîtrise des agents et mise en pratique autonome

- Personnalisation des agents IA : commandes « décorateurs », hooks et règles de cadrage.
- Atelier pratique et évaluation : réalisation autonome d'une feature par les participants avec leurs agents IA (Claude Code ou équivalent), intégrant règles métier, navigation et View Models, dans l'architecture et la solution de gestion d'état de leur choix.
- Gestion des erreurs : comparaison des types Either (Purify-ts, Effect-TS) et des exceptions JavaScript (throw / try-catch), jusqu'aux états d'erreur des View Models.
- Échanges sur les problématiques d'entreprise : micro-frontends ou frontend modulaire, agilité et pratiques Software Craft.
- Argumentation pour promouvoir le Software Craft dans les équipes, notamment face au « vibe coding ».
- Évaluation finale : QCM, auto-évaluation des capacités développées et bilan des acquis.
- Installation des agents IA, remise du projet Git et de ses différentes branches (sans state manager et avec chaque state manager).
- Mise en place du canal Slack privé pour le suivi post-formation.
- Option transversale : Analyse d'un frontend legacy généré sans pratiques Software Craft : logique métier dans les composants, useEffect en cascade et défauts architecturaux.
- Option transversale : Refactoring assisté par agents IA vers une architecture testable et maintenable.

Modalité d'accès

Contactez-nous par mail : contact@wealcomecompany.com

Délais d'accès

Nous pouvons programmer les formations en fonction de vos contraintes et de nos disponibilités.

Moyens d'accompagnement et d'assistance (Formation à distance)

En cas de difficulté d'accès à la plateforme de visioconférence ou pour toute question technique ou pédagogique pendant la session, le stagiaire dispose d'un point de contact unique pour obtenir une assistance immédiate :

Contact d'assistance technique et pédagogique : WealCome

E-mail : contact@wealcomecompany.com

Accessibilité handicap

Nos formations sont accessibles aux personnes en situation de handicap.

Les aspects, l'accessibilité et le type de handicap au regard des modalités pédagogiques sont à évoquer impérativement au cours de l'entretien préalable à toute contractualisation afin de pouvoir orienter ou accompagner au mieux les personnes en situation de handicap.