

Module 10 — Atelier Final

Jour 2 — Challenge + plan d'action

The Bearded Bear

Février 2026



Table des matières

Module 10 : Atelier Final (1h30)	2
Objectifs	2
1. Challenge Mixte (45min)	2
Exercice 1 : Projet Existant (20min)	2
Exercice 2 : Projet Vierge (25min)	3
2. Restitution (15min)	4
Partage des resultats	4
Retour d’experience collectif	5
3. Questions et Reponses (15min)	5
Questions ouvertes	5
FAQ Courantes	5
4. Plan d’Action Equipe (15min)	6
4.1 Bonnes pratiques a adopter	6
4.2 CLAUDE.md d’équipe	7
4.3 Conventions de travail avec Claude Code	7
4.4 Charte d’utilisation IA	8
4.5 Template d’onboarding nouveau membre	8
5. Quiz de Validation (10 questions)	9
Instructions	9
Reponses	11
6. Auto-Evaluation des Competences	12
7. Feedback	12
Evaluation de la formation	12
Questions ouvertes	13
Cloture	13
Certificat	13
Prochaines etapes	13
Ressources	13
Message final	14

Module 10 : Atelier Final (1h30)

Objectifs

A la fin de cet atelier, vous aurez : - Mis en pratique l'ensemble des competences acquises sur les 2 jours
- Realise un exercice complet sur un projet existant ET un projet vierge - Partage vos apprentissages avec le groupe - Defini un plan d'action pour votre equipe - Valide vos competences via un quiz

1. Challenge Mixte (45min)

Exercice 1 : Projet Existant (20min)

Contexte Vous recevez un projet existant avec du code legacy. Votre mission est d'utiliser Claude Code pour l'auditer, proposer un plan de refactoring et documenter l'existant.

Etapes 1. Audit du codebase (7min)

Demandez a Claude Code d'analyser le projet :

Analyse ce projet en profondeur :

1. Identifie l'architecture actuelle (couches, patterns, structure)
2. Detecte les anti-patterns (God Class, Long Method, Feature Envy, etc.)
3. Evalue la couverture de tests existante
4. Cherche les vulnerabilites de securite (secrets en dur, injections, etc.)
5. Calcule les metriques de complexite

Donne un rapport structure avec des scores par categorie.

2. Plan de refactoring (7min)

Utilisez le plan mode pour elaborer une strategie :

En mode plan, propose un plan de refactoring pour ce projet :

1. Priorise les problemes par impact (critique > majeur > mineur)
2. Propose un ordre de refactoring (quick wins d'abord)
3. Pour chaque refactoring, estime l'effort et le risque
4. Identifie les tests a ecrire AVANT de refactorer

Ne modifie aucun fichier, propose uniquement le plan.

3. Documentation (6min)

Demandez a Claude de generer la documentation manquante :

Genere la documentation suivante pour ce projet :

1. Un README.md avec les instructions d'installation et de demarrage
2. Un schema d'architecture en Mermaid
3. La liste des endpoints API avec leurs parametres

Base-toi uniquement sur le code existant, ne suppose rien.

Criteres d'evaluation

Critere	Points	Description
Completude de l'audit	/5	Tous les aspects sont couverts
Pertinence du plan	/5	Le plan est realiste et priorise
Qualite de la documentation	/5	La doc est exacte et utile
Utilisation des outils	/5	Sub-agents, plan mode, etc.
Total	/20	

Exercice 2 : Projet Vierge (25min)

Contexte Vous devez creer un nouveau micro-service de gestion de taches (todo-list API) en utilisant Claude Code pour le scaffolding, l'implementation et les tests.

Etapes 1. Scaffolding (8min)

Demandez a Claude de creer la structure du projet :

Cree un projet de todo-list API avec :

- Architecture Clean Architecture (domain, application, infrastructure)
- Endpoint REST : GET /tasks, POST /tasks, PUT /tasks/:id, DELETE /tasks/:id
- Modele Task : id, title, description, status (pending/done), createdAt, updatedAt
- Validation des inputs
- Gestion d'erreurs structuree

Utilise le langage/framework de ton choix adapte au contexte de formation.

2. Implementation d'une feature complete (10min)

Suivez le cycle TDD :

Implemente la fonctionnalite "filtrer les taches par statut" en TDD :

1. RED : Ecris d'abord les tests pour :
 - GET /tasks?status=pending retourne uniquement les taches pending
 - GET /tasks?status=done retourne uniquement les taches done
 - GET /tasks?status=invalid retourne une erreur 400
 - GET /tasks sans filtre retourne toutes les taches
2. GREEN : Implemente le code minimal pour faire passer les tests
3. REFACTOR : Ameliore le code en gardant les tests verts

3. Tests et qualite (7min)

Pour le projet cree :

1. Verifie que tous les tests passent
2. Ajoute les tests manquants pour atteindre 80% de couverture
3. Lance un audit de securite rapide
4. Genere un message de commit Conventional Commits pour les changements

Criteres d'evaluation

Critere	Points	Description
Architecture propre	/5	Separation des couches respectee
Tests TDD	/5	Cycle Red-Green-Refactor respecte
Qualite du code	/5	SOLID, KISS, DRY appliques
Feature fonctionnelle	/5	Le filtrage fonctionne correctement
Total	/20	

2. Restitution (15min)

Partage des resultats

Chaque participant (ou binome) presente brievement :

1. **Ce qui a bien fonctionne** (2min)
 - Quelle approche avec Claude Code a ete la plus efficace?
 - Quel outil ou pattern a ete le plus utile?

- Un moment “aha” pendant l’exercice ?
- 2. **Les difficultés rencontrées** (2min)
 - Ou Claude Code a-t-il eu du mal ?
 - Quels prompts ont nécessité des ajustements ?
 - Comment avez-vous résolu les blocages ?
- 3. **Apprentissage principal** (1min)
 - La chose la plus importante apprise pendant la formation

Retour d’expérience collectif

Le formateur synthétise les retours communs : - Patterns de prompts efficaces identifiés par le groupe
- Pièges courants à éviter - Bonnes pratiques émergentes

3. Questions et Réponses (15min)

Questions ouvertes

Temps libre pour poser des questions sur tous les sujets couverts pendant la formation.

FAQ Courantes

“Claude peut-il remplacer un développeur ?” **Non.** Claude Code est un **assistant**, pas un remplacement. Il excelle pour : - Accélérer les tâches répétitives - Générer du boilerplate - Explorer du code inconnu - Proposer des solutions

Mais il a besoin d’un développeur pour : - Définir les spécifications métier - Prendre les décisions architecturales - Valider la qualité et la pertinence du code - Comprendre le contexte business

Analogie : Claude Code est comme un copilote très compétent. Vous restez le pilote qui décide de la destination et de la route.

“Que faire quand Claude hallucine ?” Les hallucinations (réponses incorrectes mais confiantes) sont un risque réel :

Stratégies de mitigation :

1. TOUJOURS vérifier avec des tests (TDD)
2. Demander des sources/références
3. Utiliser le plan mode avant d’implémenter
4. Croiser avec la documentation officielle
5. Ne pas faire confiance aveuglément aux commandes générées

“Quel est le cout d'utilisation ?”

Profil	Consommation typique	Cout mensuel approximatif
Developpeur occasionnel	1-2h/jour	20-50 USD
Developpeur quotidien	4-6h/jour	50-150 USD
Power user (agents, CI/CD)	8h+/jour	150-400 USD

Optimisations : choix du modele, /clear, sub-agents, prompts precis.

“Comment convaincre mon equipe ?”

1. Commencer petit : un developpeur pilote pendant 2 semaines
2. Mesurer : temps gagne, qualite du code, satisfaction
3. Partager : demo des resultats concrets
4. Standardiser : CLAUDE.md d'equipe, commandes partagees
5. Former : cette formation pour toute l'equipe

“Quelles sont les limites de Claude Code ?”

Limite	Description	Mitigation
Fenetre de contexte	200K-1M tokens selon le modele	/clear, sub-agents
Hallucinations	Reponses incorrectes	Tests, verification
Cout	Facturation au token	Optimisation modele
Latence	Temps de reponse variable	Prompts precis
Code propriétaire	Le code est envoye a Anthropic	Verifier la politique de confidentialite
Dependance	Risque de sur-dependance	Garder les competences fondamentales

4. Plan d'Action Equipe (15min)**4.1 Bonnes pratiques a adopter**

Definissez ensemble les pratiques que votre equipe va adopter :

Checklist d'adoption :

- [] Créer un CLAUDE.md d'équipe avec les conventions du projet
- [] Définir les hooks de sécurité obligatoires (PreToolUse:Bash)
- [] Standardiser les slash commands de l'équipe
- [] Définir le workflow Git avec Claude Code (conventional commits, PR)
- [] Choisir le modèle par défaut (Sonnet pour le quotidien)
- [] Former tous les membres de l'équipe

4.2 CLAUDE.md d'équipe

Esquissez ensemble le contenu du CLAUDE.md de votre équipe :

```
# [Nom du Projet] - CLAUDE.md

## Architecture
- Type : [Clean Architecture / Feature-based / etc.]
- Langage : [TypeScript / Python / PHP / etc.]
- Framework : [React / FastAPI / Symfony / etc.]

## Conventions
- Tests : [Outil de test, seuil de couverture]
- Lint : [Outil, configuration]
- Format : [Outil, règles]
- Commits : Conventional Commits obligatoires

## Règles
- Toujours utiliser Docker pour les commandes
- Tests avant implementation (TDD)
- Pas de secrets dans le code
- Review obligatoire avant merge

## Commandes utiles
- `make test` : Lancer les tests
- `make lint` : Lancer le linter
- `make dev` : Démarrer l'environnement de dev
```

4.3 Conventions de travail avec Claude Code

Aspect	Convention
Modèle par défaut	Sonnet 4.6 pour le quotidien
Modèle complexe	Opus 4.6 pour architecture et refactoring majeur
Hooks obligatoires	PreToolUse:Bash (sécurité), PostToolUse:Write (lint)
Workflow Git	Conventional Commits + PR via gh

Aspect	Convention
TDD	Obligatoire pour toute nouvelle feature
Review	Code review assistee par Claude avant merge
Documentation	Mise a jour automatique avec chaque PR

4.4 Charte d'utilisation IA

Definissez les regles d'utilisation de l'IA dans votre equipe :

Charte IA - [Nom de l'equipe]

1. RESPONSABILITE

- Le developpeur est responsable du code qu'il commite, meme s'il a ete genere par Claude
- Toujours relire et comprendre le code genere

2. CONFIDENTIALITE

- Ne pas envoyer de secrets (API keys, mots de passe) a Claude
- Verifier la politique de confidentialite de l'entreprise
- Utiliser des donnees anonymisees pour les exemples

3. QUALITE

- Le code genere doit passer les memes standards que le code humain
- Tests obligatoires
- Review obligatoire

4. ETHIQUE

- Ne pas utiliser l'IA pour contourner les processus de l'equipe
- Etre transparent sur l'utilisation de l'IA
- Ne pas sur-estimer ses propres contributions

5. APPRENTISSAGE

- Utiliser Claude comme outil d'apprentissage, pas de remplacement
- Comprendre le "pourquoi" du code genere
- Maintenir et developper ses competences fondamentales

4.5 Template d'onboarding nouveau membre

Onboarding Claude Code - [Nom du Projet]

Jour 1 : Setup

- [] Installer Claude Code (Homebrew, curl, WinGet ou npm)
- [] Configurer la cle API
- [] Cloner le projet
- [] Lire le CLAUDE.md du projet
- [] Lancer ``claude`` dans le projet et explorer

Jour 2 : Decouverte

- [] Utiliser Claude pour comprendre l'architecture du projet
- [] Lancer les tests avec Claude
- [] Effectuer un petit bugfix guide par Claude
- [] Creer sa premiere PR avec Claude

Jour 3 : Autonomie

- [] Implementer une petite feature en TDD avec Claude
 - [] Utiliser les slash commands de l'equipe
 - [] Configurer ses hooks personnels
 - [] Participer a une code review assistee
-

5. Quiz de Validation (10 questions)**Instructions**

Repondez a chaque question. Score minimum pour valider : 7/10.

Question 1 : Quelle est la commande pour lancer Claude Code en mode plan?

- A) `claude --plan`
 - B) Shift+Tab pour basculer en mode plan
 - C) `/plan`
 - D) `claude -p "plan mode"`
-

Question 2 : Combien d'évenements hooks sont supportes par Claude Code?

- A) 8
 - B) 10
 - C) 13
 - D) 15
-

Question 3 : Quel exit code rend un hook bloquant ?

- A) 0
 - B) 1
 - C) 2
 - D) 255
-

Question 4 : Que fait la commande `/clear` ?

- A) Supprime tous les fichiers du projet
 - B) Reinitialise la fenetre de contexte
 - C) Efface l'historique Git
 - D) Desinstalle Claude Code
-

Question 5 : Quel est l'ordre correct du cycle TDD ?

- A) Green -> Red -> Refactor
 - B) Refactor -> Red -> Green
 - C) Red -> Green -> Refactor
 - D) Red -> Refactor -> Green
-

Question 6 : Quel outil permet de creer des sous-agents dans Claude Code ?

- A) SubAgent tool
 - B) Fork tool
 - C) Task tool
 - D) Agent tool
-

Question 7 : Quel est le protocole utilise par Claude Code pour communiquer avec des services externes ?

- A) REST
 - B) GraphQL
 - C) MCP (Model Context Protocol)
 - D) gRPC
-

Question 8 : Ou se configurent les hooks dans un projet ?

- A) `.claude/hooks.json`
 - B) `.claude/settings.json`
 - C) `CLAUDE.md`
 - D) `package.json`
-

Question 9 : Quel modele est recommande pour l'usage quotidien ?

- A) Haiku 4.5
 - B) Sonnet 4.6
 - C) Opus 4.5
 - D) Opus 4.6
-

Question 10 : Quel pattern multi-agent utilise deux agents (un qui ecrit, un qui relit) pour ameliorer la qualite ?

- A) Fan-out pattern
 - B) Interview pattern
 - C) Writer/Reviewer pattern
 - D) Leader/Follower pattern
-

Reponses

Question	Reponse
1	B
2	C
3	C
4	B
5	C
6	C
7	C
8	B
9	B

Question	Reponse
10	C

6. Auto-Evaluation des Competences

Evaluez votre niveau de confort sur chaque competence (1 = debutant, 5 = expert) :

Competence	Avant la formation	Apres la formation
Installation et configuration de Claude Code	_/5	_/5
Utilisation des commandes de base	_/5	_/5
Gestion du contexte et des tokens	_/5	_/5
Extended Thinking et prompts efficaces	_/5	_/5
Configuration de hooks	_/5	_/5
Integration MCP	_/5	_/5
Multi-agent et coordination	_/5	_/5
TDD avec Claude Code	_/5	_/5
Audit de securite	_/5	_/5
Workflow Git avec Claude	_/5	_/5

7. Feedback

Evaluation de la formation

Merci de prendre quelques minutes pour evaluer cette formation :

Aspect	1	2	3	4	5
Contenu pedagogique					
Exercices pratiques					
Rythme de la formation					
Qualite du formateur					

Aspect	1	2	3	4	5
Applicabilite au quotidien					

Questions ouvertes

1. Qu’avez-vous le plus apprecie dans cette formation ?
2. Qu’amelioreriez-vous ?
3. Quelle competence allez-vous utiliser en premier ?
4. Recommanderiez-vous cette formation a un collegue ?

Cloture

Certificat

A la fin de cette formation, chaque participant recoit un certificat attestant de la completion de la formation “Claude Code – Developpement Assiste par IA”.

Prochaines etapes

Semaine 1 : Utiliser Claude Code au quotidien avec les bases

Semaine 2 : Configurer les hooks et commandes d'equipe

Semaine 3 : Introduire le TDD avec Claude Code

Semaine 4 : Evaluer les resultats et ajuster les pratiques

Mois 2+ : Exploration avancee (multi-agent, CI/CD, Claude-Craft)

Ressources

Ressource	Lien
Documentation officielle Claude Code	docs.anthropic.com
Claude Code GitHub	github.com/anthropics/claude-code
MCP Protocol	modelcontextprotocol.io
Conventional Commits	conventionalcommits.org
OWASP Top 10	owasp.org/Top10

Ressource	Lien
Formation Claude-Craft	Contactez The Bearded CTO

Message final

Claude Code est un outil puissant qui transforme la maniere dont nous developpons. Mais comme tout outil, sa valeur depend de la maniere dont on l'utilise. Les competences acquises pendant cette formation – prompts efficaces, gestion du contexte, TDD, securite, workflows structures – sont les fondations d'une utilisation productive et responsable.

Bon developpement avec Claude Code !

Duree : 1h30 Fin de la formation Jour 2