

# Guide du Formateur

Claude Code 2.1.45 + Claude-Craft 7.13.0 — Notes et consignes

The Bearded Bear

Février 2026



## Table des matières

<b>Guide du Formateur - Claude Code 2.1.45 + Claude-Craft 7.13.0</b>	<b>2</b>
Vue d'ensemble . . . . .	2
Preparation (J-7) . . . . .	2
Checklist materiel . . . . .	2
Checklist technique . . . . .	2
Communication participants (J-3) . . . . .	2
Jour 1 : Fondamentaux & Nouveaux Projets . . . . .	3
Module 1 : Introduction a Claude Code 2.1.45 (1h30) . . . . .	3
Module 2 : Le Framework Claude-Craft 7.13.0 (1h30) . . . . .	4
Module 3 : Workflow de Developpement (2h) . . . . .	5
Module 4 : Nouveau Projet Symfony (2h) . . . . .	6
Fin de journee 1 . . . . .	7
Jour 2 : Projets Existants & Pratique Avancee . . . . .	7
Ouverture Jour 2 . . . . .	7
Module 5 : Projet Existant (1h30) . . . . .	7
Module 6 : Qualite et Securite (1h) . . . . .	8
Module 7 : Agents Specialises, BMAD et Docker (1h30) . . . . .	8
Module 8 : Outils Avances, Ralph et Autonomie (1h30) . . . . .	10
Module 9 : Atelier Pratique (1h30) . . . . .	13
Cloture . . . . .	14
Gestion des problemes courants . . . . .	15
Technique . . . . .	15
Pedagogique . . . . .	15
Quiz de fin de formation (25 questions) . . . . .	15
Fondamentaux (1-5) . . . . .	15
Hooks et Outils (6-10) . . . . .	15
Skills et Agents (11-14) . . . . .	16
BMAD v6 (15-18) . . . . .	16
Ralph (19-20) . . . . .	16
QA Recette (22-23) . . . . .	16
Qualite et Securite (24-25) . . . . .	16
Reponses attendues . . . . .	16
Scripts de demo prepares . . . . .	17
Script Ralph Wiggum . . . . .	17
Script QA Recette . . . . .	18
Contacts . . . . .	18

## Guide du Formateur - Claude Code 2.1.45 + Claude-Craft 7.13.0

### Vue d'ensemble

Ce guide fournit les instructions détaillées pour animer la formation de 2 jours sur Claude Code 2.1.45 et Claude-Craft 7.13.0.

**Versions couvertes :** - Claude Code : 2.1.45 (hooks, plan mode, background tasks, keybindings, extended thinking, MCP) - Claude-Craft : 7.13.0 (TCL, npx install, 10 stacks, BMAD v6, Ralph, QA Recette, 33 agents, 160 commands)

---

### Preparation (J-7)

#### Checklist materiel

- ☐ Salle avec videoprojecteur
- ☐ WiFi stable et rapide
- ☐ Acces electrique pour tous les postes
- ☐ Tableau blanc ou paperboard
- ☐ Post-its pour les retours

#### Checklist technique

- ☐ Node.js 20+ installe sur le poste formateur
- ☐ Claude Code 2.1.45 installe et fonctionnel
- ☐ Projet de demo prepare avec Claude-Craft 7.13.0
- ☐ Compte Anthropic de demo avec credits
- ☐ Slides exportes en PDF (backup)
- ☐ Exercices testes sur la derniere version
- ☐ Chrome avec extension claude-in-chrome (v1.0.36+) pour demos QA Recette
- ☐ Docker installe et fonctionnel pour les demos

### Communication participants (J-3)

Envoyer un email avec : - Prerequis techniques (Node.js 20+, Git, IDE) - Demande de creer un compte Anthropic - Lien vers la documentation Claude Code - Programme resume

---

## Jour 1 : Fondamentaux & Nouveaux Projets

### Module 1 : Introduction a Claude Code 2.1.45 (1h30)

#### Timing detaille

Duree	Contenu	Support
5 min	Accueil, tour de table	-
15 min	Presentation Claude Code 2.1.45	Slides
20 min	Demo installation live	Terminal
15 min	Commandes de base + nouvelles	Demo
20 min	Plan Mode et Background Tasks	Demo
15 min	<b>Exercice 1</b> : Premier projet	Exercice

#### Points cles a insister (2.1.45)

1. **Plan Mode** : Montrer `/plan` pour exploration securisee
2. **Background Tasks** : Montrer `/tasks` pour operations longues
3. **Hooks** : Mentionner les 13 evenements disponibles (PreToolUse, PostToolUse, PostToolUse-Failure, PermissionRequest, UserPromptSubmit, Stop, SubagentStop, SubagentStart, Notification, PreCompact, SessionStart, SessionEnd, Setup) 3b. **Task Management System** : Task-Create/Get/Update/List pour gestion multi-taches 3c. **/keybindings** : Personnalisation des raccourcis clavier
4. **Cost des tokens** : Montrer `/cost` regulierement
5. **Extended Thinking** : Mentionner la hierarchie de raisonnement

#### Demo Plan Mode

```
# Activer le mode plan
/plan
```

```
# Claude peut lire et analyser, mais pas modifier
"Analyse la structure de ce projet et propose un plan de refactoring"
```

```
# Claude analyse sans modifier...
```

```
# Quitter le mode plan
/plan off
```

**Nouveaute v7.13.0** : Les commandes Claude-Craft classifient automatiquement le plan mode requis (MANDATORY / RECOMMENDED / CONDITIONAL). Montrer un exemple : `/work-`

`flow:implement` active automatiquement le plan mode, tandis que `/workflow:status` ne l'active jamais.

### Questions frequentes

“Quelle difference avec ChatGPT ?”

Claude Code est specialise pour le developpement, avec acces au filesystem, hooks pour automatisation, plan mode pour exploration, background tasks pour operations longues, extended thinking pour le raisonnement complexe, et MCP pour les integrations.

“C'est cher ?”

Montrer le calcul : ~0.03 par interaction simple. Formation de 2 jours = 5-10 de tokens.

## Module 2 : Le Framework Claude-Craft 7.13.0 (1h30)

### Timing detaille

Duree	Contenu	Support
20 min	TCL (Tiered Context Loading)	Slides + Schema
20 min	Nouvelle structure et 10 stacks	Slides
15 min	Skills et References	Demo
15 min	Installation via npx	Terminal
20 min	<b>Exercice 2</b> : Installation	Exercice

### Points cles a insister (7.13.0)

1. **TCL** = 95% d'economie de tokens (dessiner le schema au tableau)
2. **NPX** = Plus de git clone + make
3. **10 stacks** = Multi-technologie
4. **context.yaml** = Chargement automatique
5. **BMAD v6** = Gestion de projet integree
6. **Ralph** = Boucle continue autonome
7. **QA Recette** = Tests d'acceptation automatises

### Demo TCL + Installation

```
# Installation via npx
npx @the-bearded-bear/claude-craft install

# Ou directement
npx @the-bearded-bear/claude-craft install ~/demo --tech=symfony --lang=fr

# Explorer la structure TCL
ls -la ~/demo/.claude/

# Montrer le CLAUDE.md minimal (~700 bytes)
cat ~/demo/.claude/CLAUDE.md

# Montrer l'INDEX.md
cat ~/demo/.claude/INDEX.md

# Montrer le context.yaml
cat ~/demo/.claude/context.yaml
```

### Comparaison visuelle

```
+-----+
| AVANT (v3.x)                                |
| ~70,000 tokens auto-charges                 |
| CLAUDE.md de ~15,000 caracteres             |
+-----+
| APRES (v7.13.0 TCL)                        |
| ~3,500 tokens auto-charges                 |
| CLAUDE.md de ~700 bytes                   |
| = 95% d'economie !                        |
+-----+
```

## Module 3 : Workflow de Developpement (2h)

### Timing detaille

Duree	Contenu	Support
30 min	Les 3 tracks (Quick Flow, Standard, Enterprise)	Slides + Tableau
40 min	Commandes workflow (demo)	Terminal
20 min	Workflow d'analyse obligatoire	Slides
30 min	<b>Exercice 3</b> : Workflow Standard	Exercice

---

Duree	Contenu	Support
-------	---------	---------

---

### Demo du workflow avec Plan Mode

# 1. Activer le mode plan pour explorer  
/plan

# 2. Analyser avec le workflow  
/workflow:init "Ajouter un champ phone a l'entite User"

# 3. Claude analyse sans modifier  
/workflow:analyze

# 4. Proposer un plan  
/workflow:plan

# 5. Quitter le mode plan pour executer  
/plan off

# 6. Implementer  
/workflow:implement

---

## Module 4 : Nouveau Projet Symfony (2h)

### Timing detaille

Duree	Contenu	Support
20 min	Configuration initiale	Demo
30 min	Generation de features	Demo
20 min	Architecture Clean	Slides + Schema
50 min	<b>Exercice 4</b> : Projet from scratch	Exercice

---

### Points cles a insister

1. **Clean Architecture** : Dessiner le schema au tableau
  2. **Generation != Magie** : Toujours valider le code genere
  3. **TDD** : Montrer le cycle Red-Green-Refactor
-

**Fin de journée 1**

Duree	Contenu
10 min	Recapitulatif TCL + Plan Mode
5 min	Questions/reponses
5 min	Aperçu du jour 2 (Hooks, BMAD, Ralph, QA Recette)
5 min	Devoirs (optionnel) : explorer les skills

**Jour 2 : Projets Existants & Pratique Avancee****Ouverture Jour 2**

Duree	Contenu
10 min	Retour sur les questions du J1
5 min	Programme de la journée

**Module 5 : Projet Existant (1h30)****Timing detaille**

Duree	Contenu	Support
30 min	Commandes d'audit (demo)	Terminal
15 min	Interpretation des rapports	Slides
15 min	Strategie d'adoption	Slides
30 min	<b>Exercice 5</b> : Audit projet	Exercice

**Demo de l'audit avec Plan Mode**

```
# Utiliser le code legacy prepare
cd ~/legacy-demo
claude
```



```
# Activer le mode plan pour audit sans modification
/plan

/symfony:check-compliance
# Commenter chaque section du rapport

/symfony:check-security
# Montrer les vulnerabilites

# Quitter le plan mode pour corriger
/plan off
# Montrer une correction live
```

---

## Module 6 : Qualite et Securite (1h)

### Timing detaille

Duree	Contenu	Support
15 min	Pre-commit checklist	Slides
15 min	TDD avec Claude (skill /testing)	Demo
20 min	OWASP Top 10 (skill /security)	Slides + Exemples
10 min	<b>Exercice 6</b> : Audit qualite	Exercice

### Demo Skills

```
# Charger le skill testing
/testing

# Claude repond maintenant avec contexte TDD
"Ecris les tests pour OrderService"

# Charger le skill security
/security

# Claude repond avec contexte OWASP
"Analyse ce code de paiement"
```

---

## Module 7 : Agents Specialises, BMAD et Docker (1h30)

### Timing detaille

Duree	Contenu	Support
20 min	Skills system (2.1.45)	Slides + Demo
15 min	Panorama des agents	Slides
15 min	Agents BMAD v6 (demo)	Terminal
10 min	Agents Docker (demo)	Terminal
10 min	Invocation et contexte (demo)	Terminal
20 min	<b>Exercice 7</b> : Code review	Exercice

### Demo Skills + Agents

```
# Montrer les skills disponibles  
/skills
```

```
# Charger un skill  
/testing
```

```
# Utiliser un agent avec le skill charge  
"Agis comme @tdd-coach et aide-moi a ecrire les tests"
```

```
# Montrer context.yaml  
cat .claude/context.yaml
```

### Demo Workflow BMAD v6

```
# Initialiser le workflow  
/workflow:init
```

```
# Montrer le status du projet  
/workflow:status
```

```
# Utiliser le Product Owner  
"Agis comme @product-owner et redige la vision produit"
```

```
# Utiliser le Tech Lead  
"Agis comme @tech-lead et valide l'architecture"
```

```
# Valider une quality gate  
/gate:validate-prd  
# Montrer le seuil de 80% et les criteres
```

### Demo Agents Docker

```
# Architecture Docker  
"Agis comme @docker-architect et propose l'architecture"
```

```
# Generer docker-compose
```

```
/docker:compose-setup
```

```
# Debug un container
```

```
"Agis comme @docker-debug et diagnostique ce probleme"
```

## Module 8 : Outils Avances, Ralph et Autonomie (1h30)

### Timing detaille

Duree	Contenu	Support
15 min	Hooks (13 evenements)	Slides + Demo
15 min	Ralph Wiggum (demo)	Terminal
15 min	Extended Thinking + MCP	Slides + Demo
15 min	QA Recette (demo)	Chrome + Terminal
15 min	Permissions + Plugins	Demo

### Les 13 evenements de Hooks

Evenement	Declencheur	Usage typique
<b>PreToolUse</b>	Avant l'utilisation d'un outil	Valider/bloquer une action avant execution
<b>PostToolUse</b>	Apres succes d'un outil	Linter, formater apres ecriture
<b>PostToolUseFailure</b>	Apres echec d'un outil	Auto-recovery, logging d'erreurs
<b>PermissionRequest</b>	Demande de permission	Auto-approuver certaines actions
<b>UserPromptSubmit</b>	Soumission d'un prompt utilisateur	Enrichir le prompt, ajouter du contexte
<b>Stop</b>	Fin de reponse Claude	Verifications finales, notifications
<b>SubagentStop</b>	Arret d'un sous-agent	Valider le travail du sous-agent
<b>SubagentStart</b>	Lancement d'un sous-agent	Logging, configuration contexte
<b>Notification</b>	Notification systeme	Alertes, logging
<b>PreCompact</b>	Avant la compaction du contexte	Sauvegarder des infos avant compression

Evenement	Declencheur	Usage typique
<b>SessionStart</b>	Debut/reprise de session	Initialisation, chargement de contexte
<b>SessionEnd</b>	Fin de session	Nettoyage, sauvegarde
<b>Setup</b>	Premier lancement (-init, -maintenance)	Configuration premiere utilisation

## Demo Hooks

*# Montrer la configuration des hooks*

```
cat .claude/settings.json
```

*# Exemple de configuration avec les vrais evenements*

```
{
  "hooks": {
    "PostToolUse": [{
      "matcher": "Write",
      "command": "php vendor/bin/php-cs-fixer fix $FILE_PATH --quiet"
    }],
    "PreToolUse": [{
      "matcher": "Bash",
      "command": "echo 'Verification avant execution...'"
    }],
    "Stop": [{
      "command": "echo 'Session terminee' >> /var/log/claude.log"
    }],
    "SessionStart": [{
      "command": "echo 'Bienvenue! Chargement du contexte...'"
    }]
  }
}
```

*# Montrer un hook en action*

```
"Modifie UserController pour ajouter une validation"
```

*# Le hook PostToolUse s'execute automatiquement apres*

## Demo Ralph Wiggum

*# Lancer Ralph sur une tache*

```
/common:ralph-run "Implementer l'authentification utilisateur"
```

*# Ralph execute en boucle continue :*

*# 1. Analyse la tache*

*# 2. Execute les etapes*

*# 3. Valide la Definition of Done (DoD)*

*# 4. Recommence si DoD non atteinte*

```
# Les 5 types de validateurs DoD :  
# - command : Executer tests, lint, build  
# - output_contains : Verifier des patterns dans la sortie  
# - file_changed : Verifier les modifications de fichiers  
# - hook : Integrer avec quality-gate.sh  
# - human : Approbation manuelle  
  
# Circuit breaker : arret automatique apres N echecs consecutifs  
# pour eviter les boucles infinies
```

### Demo Extended Thinking + MCP

```
# Extended Thinking - hierarchie de raisonnement  
# Claude utilise un processus de reflexion en couches :  
# 1. Reflexion rapide (reponses simples)  
# 2. Reflexion approfondie (architecture, design)  
# 3. Reflexion etendue (problemes complexes multi-fichiers)  
  
# Montrer Extended Thinking en action  
"Refactore ce module en Clean Architecture avec CQRS"  
# Observer le processus de reflexion etendue  
  
# MCP (Model Context Protocol) - integrations  
# Permet a Claude de se connecter a des outils externes :  
# - Navigateur Chrome (claude-in-chrome)  
# - Bases de donnees  
# - APIs externes  
# - Outils de monitoring
```

### Demo QA Recette

```
# Prerequis : Chrome avec extension claude-in-chrome v1.0.36+  
# Lancer Claude avec le flag --chrome ou /chrome  
  
# Tester une story specifique  
/qa:recette --scope=story --id=US-001  
  
# Dry run pour voir le plan de test  
/qa:recette --scope=story --id=US-001 --dry-run  
  
# Tester un sprint complet  
/qa:recette --scope=sprint --id=Sprint-3  
  
# Reprendre une session interrompue  
/qa:recette --resume=REC-20260130-143022  
  
# La Golden Rule : un bug corrige ne doit JAMAIS reapparaître  
# Chaque bug detecte genere automatiquement un test de regression
```

```
# Structure de sortie :  
# .recette/  
# +-- plans/           -> Plans de test (YAML)  
# +-- sessions/        -> Etats de session  
# +-- regression/      -> Suite de regression  
# +-- metrics/         -> Donnees historiques  
# +-- reports/         -> Rapports generes
```

### Demo Background Tasks

```
# Lancer une tache en background  
/tasks run "Genere la documentation API complete"  
  
# Lister les taches  
/tasks list  
  
# Continuer a travailler  
"Aide-moi avec autre chose..."  
  
# Verifier le statut plus tard  
/tasks status task-123
```

---

## Module 9 : Atelier Pratique (1h30)

### Timing detaille

Duree	Contenu	Support
30 min	Challenge en binomes	Exercice 8
15 min	Demo BMAD sprint complet	Terminal
15 min	Demo QA Recette live	Chrome + Terminal
15 min	Presentations (2 min/binome)	-
15 min	Discussion collective	-

### Demo BMAD Sprint Complet

```
# Montrer le cycle complet BMAD  
/workflow:init  
  
# 1. Creer la vision produit avec @pm  
"Agis comme @pm et cree la vision pour une app e-commerce"
```

```
# 2. Valider la PRD (quality gate >= 80%)
/gate:validate-prd

# 3. Design technique avec @architect
"Agis comme @architect et redige la spec technique"

# 4. Generer le backlog avec @po
"Agis comme @po et cree le backlog du sprint"

# 5. Executer le sprint
/project:run-sprint

# 6. Montrer le statut
/workflow:status
```

### Demo QA Recette Live

```
# Preparer la demo avec le projet existant
# S'assurer que Chrome est ouvert avec l'extension

# 1. Generer le plan de test
/qa:recette --scope=story --id=US-001 --dry-run

# 2. Executer les tests en live
/qa:recette --scope=story --id=US-001

# 3. Montrer la detection de regression
/qa:regression

# 4. Generer le rapport
/qa:report
```

---

### Cloture

---

Duree	Contenu
10 min	Quiz de validation (optionnel)
5 min	Auto-evaluation
10 min	Questions finales
5 min	Remise des certificats
5 min	Prochaines etapes et ressources

---

## Gestion des problemes courants

### Technique

Probleme	Solution
Cle API invalide	Avoir des cles backup
npx echoue	git clone + npm link comme fallback
Quota depasse	Changer de compte
Claude ne repond pas	Verifier connexion, relancer
Structure TCL differente	Verifier version Claude-Craft (7.x+)
Extension Chrome non detectee	Verifier version $\geq 1.0.36$ , relancer Chrome
Docker non disponible	Installer Docker Desktop, verifier le daemon

### Pedagogique

Probleme	Solution
Niveau heterogene	Former des binomes mixtes
Questions hors sujet	“Parking lot” au tableau
Retard sur le planning	Reduire les exercices, pas le contenu
Participant desengage	Lui donner un role actif

## Quiz de fin de formation (25 questions)

### Fondamentaux (1-5)

1. Qu'est-ce que le TCL et quelle economie apporte-t-il?
2. Quelle commande pour activer le Plan Mode?
3. Comment installer Claude-Craft via npx?
4. Que contient le fichier context.yaml?
5. Combien de stacks et d'agents supporte Claude-Craft 7.13.0?

### Hooks et Outils (6-10)

6. Citez les 13 evenements de hooks disponibles dans Claude Code 2.1.45.



7. Quelle est la difference entre PreToolUse et PostToolUse?
8. Comment lancer une tache en background?
9. Comment personnaliser les keybindings?
10. Qu'est-ce que l'Extended Thinking et quelle est sa hierarchie de raisonnement?

### **Skills et Agents (11-14)**

11. Difference entre Skill et Agent?
12. Comment charger le skill testing?
13. Citez 2 agents projet et leur role.
14. Quel agent utiliser pour diagnostiquer un probleme Docker?

### **BMAD v6 (15-18)**

15. Quels sont les 3 tracks de developpement BMAD et quand les utiliser?
16. Quels sont les seuils des quality gates (PRD, Tech Spec, Backlog, Sprint Ready, Story DoD)?
17. Quels sont les 2 agents projet disponibles et ou sont integres les roles BMAD?
18. Quelle commande pour valider une PRD et quel est le seuil minimum?

### **Ralph (19-20)**

19. Quels sont les 5 types de validateurs DoD de Ralph?
20. Qu'est-ce que le circuit breaker de Ralph et a quoi sert-il?

### **QA Recette (22-23)**

22. Qu'est-ce que la Golden Rule du QA Recette?
23. Quelle extension Chrome est requise pour le QA Recette et quelle version minimum?

### **Qualite et Securite (24-25)**

24. Que signifie TDD et quelles sont les 3 phases du cycle?
25. Citez 3 vulnerabilites OWASP Top 10.

### **Reponses attendues**

1. Tiered Context Loading, ~95% d'economie de tokens
2. /plan
3. `npx @the-bearded-bear/claude-craft install`
4. Les references a charger selon le contexte du projet
5. 10 stacks, 33 agents

6. PreToolUse, PostToolUse, PostToolUseFailure, PermissionRequest, UserPromptSubmit, Stop, SubagentStop, SubagentStart, Notification, PreCompact, SessionStart, SessionEnd, Setup
  7. PreToolUse se declenche avant l'execution d'un outil (peut bloquer), PostToolUse apres (peut formater/valider)
  8. `/tasks run "description"`
  9. Via le fichier de configuration keybindings dans settings
  10. Processus de reflexion en couches : rapide, approfondie, etendue selon la complexite
  11. Skill = contexte/connaissances chargees, Agent = role avec expertise specifique
  12. `/testing`
  13. `@product-owner` (Product management, CSPO), `@tech-lead` (Technical leadership)
  14. `@docker-debug`
  15. Quick Flow (< 5min, bug fixes), Standard (< 15min, features), Enterprise (< 30min, platforms)
  16. PRD >= 80%, Tech Spec >= 90%, Backlog INVEST 6/6, Sprint Ready 100%, Story DoD 100%
  17. `@product-owner` et `@tech-lead`. Les roles BMAD (pm, ba, architect, po, sm, dev, qa, ux) sont integres dans les commandes `/workflow` : et `/sprint` :, pas en standalone
  18. `/gate:validate-prd`, seuil minimum de 80%
  19. `command`, `output_contains`, `file_changed`, `hook`, `human`
  20. Mecanisme d'arret automatique apres N echecs consecutifs pour eviter les boucles infinies
  21. Un bug corrige ne doit JAMAIS reapparaître - chaque bug genere automatiquement un test de regression
  22. `claude-in-chrome`, version minimum 1.0.36
  23. Test-Driven Development, phases : Red (ecrire test qui echoue) -> Green (code minimal pour passer) -> Refactor (ameliorer)
  24. Injection SQL, XSS (Cross-Site Scripting), CSRF, Broken Authentication, Security Misconfiguration, etc.
- 

## Scripts de demo prepares

### Script Ralph Wiggum

```
# Preparer un projet avec une tache claire
# Objectif : montrer la boucle continue

# 1. Definir la tache
/common:ralph-run "Ajouter la validation email sur le formulaire
  ↳ d'inscription"

# 2. Observer Ralph :
#   - Analyse le code existant
#   - Ecrit les tests (TDD Red)
#   - Implemente la validation (TDD Green)
#   - Refactore si necessaire (TDD Refactor)
#   - Valide la DoD
```

```
# - Recommence si echec

# 3. Points a commenter pendant la demo :
# - La boucle continue (pas besoin d'intervention)
# - Les validateurs DoD qui verifient le travail
# - Le circuit breaker qui protege contre les boucles infinies
# - L'integration avec les quality gates BMAD
```

## Script QA Recette

```
# Prerequis : extension Chrome claude-in-chrome >= 1.0.36
# Preparer une story avec criteres d'acceptation clairs

# 1. Dry run pour montrer le plan
/qa:recette --scope=story --id=US-DEMO --dry-run

# 2. Commenter le plan genere :
# - Tests derives des criteres d'acceptation
# - Scenarios nominaux et d'erreur
# - Checkpoints pour la reprise

# 3. Executer les tests
/qa:recette --scope=story --id=US-DEMO

# 4. Observer dans Chrome :
# - Navigation automatique
# - Interactions avec les formulaires
# - Captures d'ecran automatiques
# - Detection d'anomalies

# 5. Montrer la Golden Rule :
# - Simuler un bug
# - Montrer le test de regression genere
# - Expliquer que ce test sera rejoue a chaque recette

# 6. Generer le rapport
/qa:report
```

---

## Contacts

- Support technique : [email]
  - Questions pedagogiques : [email]
  - Urgence : [telephone]
-

**Version formation : 3.0.0 Claude Code : 2.1.45 Claude-Craft : 7.13.0**

**Bonne formation !**