

Module 3 — Workflow de Développement

Jour 1 — Méthodologie et bonnes pratiques

The Bearded Bear

Février 2026



Table des matières

Module 3 : Workflow de Développement et BMAD v6	2
Objectifs	2
1. Les 3 Tracks de Workflow	2
Vue d'ensemble	2
Track Quick Flow (< 5 min)	2
Track Standard (< 15 min)	2
Track Enterprise (< 30 min)	3
2. Commandes Workflow	3
/workflow :init	3
/workflow :analyze	3
/workflow :plan	4
/workflow :design	4
/workflow :implement	5
Plan Mode intégré aux commandes	5
3. Workflow d'Analyse Obligatoire	5
Principe Fondamental	5
Les 4 Étapes	6
Critères de Décision	6
Exemple d'Analyse	7
4. BMAD v6 Quality Gates	7
Principe	7
Les 5 Quality Gates	8
Détail des Gates	8
Commandes de Validation	8
5. Status Routing (State Machine)	9
Machine à États	9
Phases TDD dans le Status Routing	9
Commandes Sprint	9
Transitions Valides	10
6. Cycle TDD avec Claude	10
Red-Green-Refactor	10
Avec Claude	10
Atelier Pratique	11
Scénario	11
Étapes	11
Critères de succès	11
Points Clés à Retenir	12

Module 3 : Workflow de Développement et BMAD v6

Objectifs

À la fin de ce module, vous serez capable de : - Choisir le workflow adapté à votre tâche - Utiliser les commandes workflow - Appliquer le workflow d'analyse obligatoire - Suivre le cycle TDD avec Claude - Comprendre BMAD v6 et ses quality gates - Maîtriser le status routing

1. Les 3 Tracks de Workflow

Vue d'ensemble

Track	Setup	Phases	Best For
Quick Flow	< 5 min	Implement only	Bug fixes, hotfixes
Standard	< 15 min	Plan → Design → Implement	New features
Enterprise	< 30 min	Analyze → Plan → Design → Implement	Platforms

Track Quick Flow (< 5 min)

Quand l'utiliser : - Bugfix simple - Hotfix en production - Correction de typo - Petite modification isolée

Étapes : 1. Implémentation directe 2. Test 3. Commit

Exemple :

```
"Corrige le bug où l'email n'est pas validé dans UserController"
# Claude analyse, corrige, propose les tests
```

Track Standard (< 15 min)

Quand l'utiliser : - Nouvelle feature - Refactoring ciblé - Ajout d'endpoint API - Intégration service externe

Étapes : 1. `/workflow:init` - Analyse et recommandation 2. `/workflow:plan` - PRD et backlog 3. `/workflow:design` - Spécifications techniques 4. `/workflow:implement` - Développement TDD 5. Code Review

Exemple :

```
/workflow:init "Ajouter un système de notifications par email"  
# Claude analyse, recommande Standard, propose les étapes
```

Track Enterprise (< 30 min)

Quand l'utiliser : - Projet complet / plateforme - Refactoring majeur - Migration technique - Nouvelle fonctionnalité majeure

Étapes : 1. /workflow:analyze - Recherche et exploration 2. /workflow:plan - PRD et backlog 3. /workflow:design - Architecture et spécifications techniques 4. /workflow:implement - Développement itératif en sprints 5. Revue d'architecture

2. Commandes Workflow

/workflow :init

But : Analyser la demande et recommander un track

```
/workflow:init "Implémenter l'authentification JWT"
```

```
# Output:  
# Analyse de la demande  
# - Complexité estimée: Moyenne  
# - Fichiers impactés: ~10  
# - Tests nécessaires: ~15  
#  
# Track recommandé: STANDARD  
#  
# Voulez-vous procéder avec ce workflow ?
```

/workflow :analyze

But : Explorer le code existant (track Enterprise)

```
/workflow:analyze
```

```
# Output:  
# Structure actuelle:  
# - src/Entity/User.php (existant)  
# - src/Controller/AuthController.php (à créer)  
#  
# Dépendances identifiées:  
# - lexik/jwt-authentication-bundle  
# - symfony/security-bundle  
#
```

```
# Points d'attention:  
# - Pas de tests existants pour l'auth
```

/workflow:plan

But : Générer PRD et backlog

/workflow:plan

```
# Output:  
# PRD - Authentification JWT  
#  
# ## User Stories  
# 1. En tant qu'utilisateur, je peux m'inscrire  
# 2. En tant qu'utilisateur, je peux me connecter  
# 3. En tant qu'utilisateur, je peux rafraîchir mon token  
#  
# ## Backlog  
# - [ ] Configurer JWT bundle  
# - [ ] Créer endpoint /register  
# - [ ] Créer endpoint /login  
# - [ ] Créer endpoint /token/refresh  
# - [ ] Ajouter tests
```

/workflow:design

But : Spécifications techniques

/workflow:design

```
# Output:  
# Design Technique  
#  
# ## Architecture  
# - Handler: RegisterUserHandler  
# - Command: RegisterUserCommand  
# - Validator: RegisterUserValidator  
#  
# ## Endpoints  
# POST /api/auth/register  
# POST /api/auth/login  
# POST /api/auth/refresh  
#  
# ## Schéma  
# User: id, email, password, roles, createdAt
```

/workflow :implement

But : Développement guidé TDD

/workflow:implement

```
# Claude va:  
# 1. Générer les tests en premier (RED)  
# 2. Implémenter le code (GREEN)  
# 3. Proposer des refactorings (REFACTOR)
```

Plan Mode intégré aux commandes

Chaque commande workflow inclut désormais une guidance Plan Mode qui indique automatiquement quand l'activer :

Commande	Plan Mode	Raison
/workflow:implement	MANDATORY	Analyse du code impacté et validation du plan avant modification
/workflow:plan	RECOMMENDED	Planification complexe multi-modules
/workflow:design	RECOMMENDED	Conception architecture nécessitant exploration
/workflow:analyze	CONDITIONAL	S'active automatiquement si le scope couvre plusieurs modules
/workflow:init	CONDITIONAL	S'active si le projet est complexe
/workflow:status	—	Lecture seule, pas de plan mode

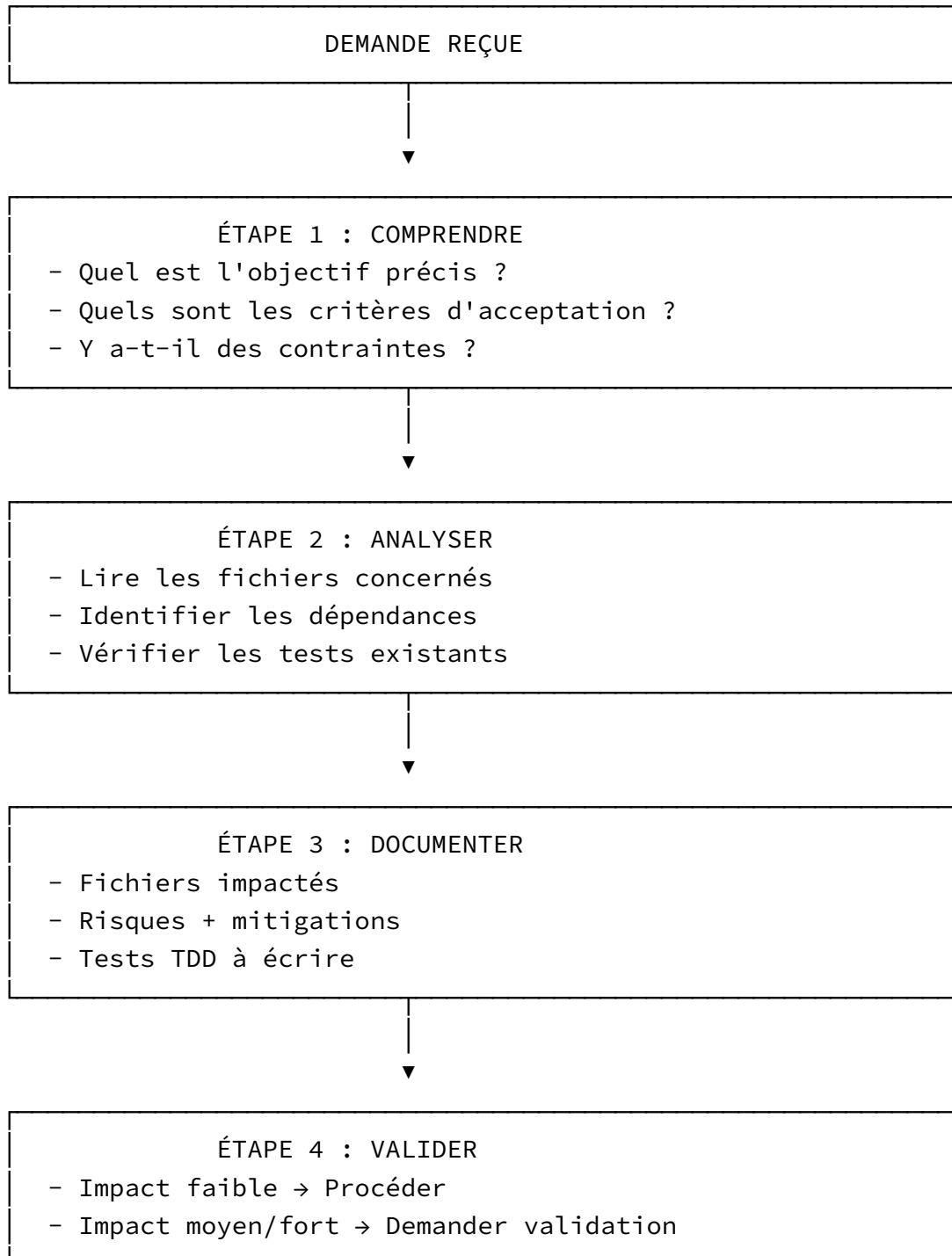
Les commandes QA suivent la même logique : /qa:tdd et /qa:recette sont **MANDATORY**, /qa:status et /qa:report ne nécessitent pas de plan mode.

3. Workflow d'Analyse Obligatoire

Principe Fondamental

AVANT toute modification de code, une phase d'analyse est OBLIGATOIRE.

Les 4 Étapes



Critères de Décision

Impact	Caractéristiques	Action
Faible	1 fichier, pas de breaking change, < 1h	Procéder directement
Moyen	2-5 fichiers, migration DB possible, < 4h	Validation recommandée
Fort	> 5 fichiers, breaking changes, refactoring archi	Validation obligatoire

Exemple d'Analyse

Analyse : Ajout de notifications email

Objectif

Envoyer une notification email lors de la création d'une commande.

Fichiers impactés

- ``OrderService`` (ajout dispatch event)
- ``NotificationListener`` (nouveau)
- ``EmailService`` (utilisation existante)
- Tests unitaires pour le listener

Impacts

- Breaking change : NON
- Migration DB : NON
- Performance : Faible (async recommandé)
- Données sensibles : Email utilisateur (déjà géré)

Risques

1. Surcharge email → Mitigation : queue async
2. Email en spam → Mitigation : configuration DKIM/SPF

Tests TDD

1. `test_should_send_email_on_order_created()`
2. `test_should_not_send_if_user_opted_out()`
3. `test_should_handle_email_failure_gracefully()`

4. BMAD v6 Quality Gates

Principe

BMAD v6 impose des **quality gates** (portes de qualité) entre chaque phase du workflow. Une phase ne peut pas commencer tant que la phase précédente n'a pas atteint le seuil requis. Cela garantit la qualité à chaque étape du développement.

Les 5 Quality Gates

Gate	Threshold	When
PRD Gate	≥80%	Vision → PRD
Tech Spec Gate	≥90%	PRD → Tech Spec
Backlog Gate	INVEST 6/6	Tech Spec → Backlog
Sprint Ready	100%	Backlog → Sprint
Story DoD	100%	Dev → Done

Détail des Gates

PRD Gate (≥80%) : Le Product Requirements Document doit couvrir au moins 80% des critères attendus (objectifs, user stories, contraintes, critères d'acceptation) avant de passer au design technique.

Tech Spec Gate (≥90%) : Les spécifications techniques doivent atteindre 90% de complétion (architecture, endpoints, schéma de données, diagrammes) avant de générer le backlog.

Backlog Gate (INVEST 6/6) : Chaque story du backlog doit respecter les 6 critères INVEST : - **I**ndependent - Indépendante des autres stories - **N**egotiable - Négociable, pas un contrat - **V**aluable - Apporte de la valeur - **E**stimable - Estimable en effort - **S**mall - Suffisamment petite - **T**estable - Testable avec des critères clairs

Sprint Ready (100%) : Toutes les stories du sprint doivent être complètement prêtes (critères d'acceptation, estimations, dépendances identifiées).

Story DoD (100%) : La Definition of Done de chaque story doit être entièrement satisfaite (code, tests, review, documentation).

Commandes de Validation

```
# Valider un PRD avant de passer au design
/gate:validate-prd
```

```
# Output:
# PRD Validation Report
# Score: 85% (threshold: 80%) - PASSED
# - Objectifs: OK
# - User Stories: OK
# - Contraintes: OK
# - Critères d'acceptation: 2 stories manquent des critères
```

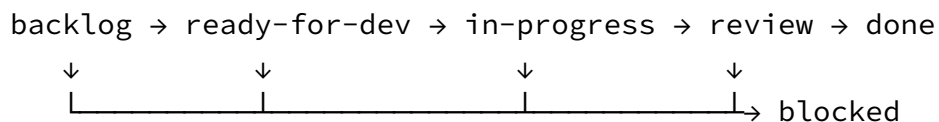
```
# Valider la Definition of Done d'une story
/gate:validate-story
```

```
# Output:
# Story DoD Validation
# Score: 100% - PASSED
# - Code implémenté: OK
# - Tests unitaires: OK (12/12 passing)
# - Tests intégration: OK (3/3 passing)
# - Code review: OK
# - Documentation: OK
```

5. Status Routing (State Machine)

Machine à États

Le status routing de BMAD v6 définit le cycle de vie de chaque story via une machine à états. Chaque story passe par des états bien définis, avec la possibilité d'être bloquée à tout moment.



Phases TDD dans le Status Routing

Quand une story est in-progress, le développement suit le cycle TDD :

RED → GREEN → REFACTOR

1. **Red** : Écrire les tests qui échouent
2. **Green** : Écrire le code minimum pour faire passer les tests
3. **Refactor** : Améliorer le code en gardant les tests verts

Commandes Sprint

```
# Récupérer la prochaine story prête à être développée
/sprint:next-story
```

```
# Output:
# Next story: US-042 "Ajouter la validation email"
# Status: ready-for-dev
# Priority: High
# Estimation: 3 points
# Acceptance criteria: 4 critères définis

# Transitionner une story vers un nouvel état
```

```
/sprint:transition
```

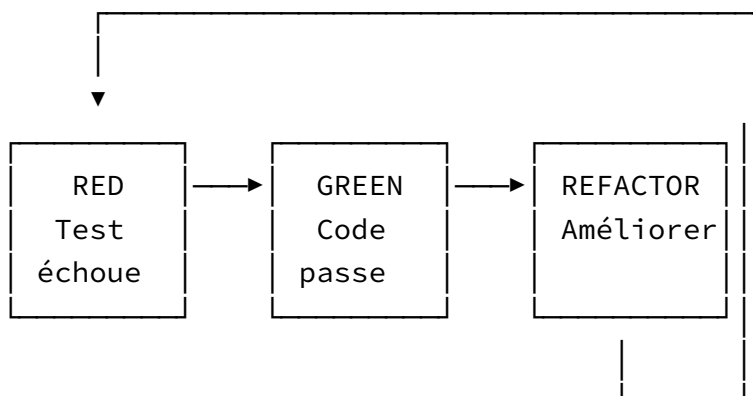
```
# Output:
# US-042: ready-for-dev → in-progress
# Developer assigned
# TDD Phase: RED (écrire les tests)
```

Transitions Valides

De	Vers	Condition
backlog	ready-for-dev	Sprint Ready gate = 100%
ready-for-dev	in-progress	Développeur assigné
in-progress	review	Story DoD gate = 100%
review	done	Code review approuvée
<i>tout état</i>	blocked	Problème bloquant identifié
blocked	<i>état précédent</i>	Problème résolu

6. Cycle TDD avec Claude

Red-Green-Refactor



Avec Claude

```
# 1. RED - Demander les tests d'abord
"Écris les tests PHPUnit pour un OrderService qui crée des commandes"

# Claude génère les tests (qui échouent car pas d'implémentation)
```

```
# 2. GREEN - Demander l'implémentation
"Maintenant implémente OrderService pour faire passer ces tests"

# Claude implémente le minimum pour passer les tests

# 3. REFACTOR - Demander l'amélioration
"Refactore ce code en appliquant les principes SOLID"

# Claude propose des améliorations en gardant les tests verts
```

Atelier Pratique

Scénario

Implémenter une feature “Ajout au panier” avec le workflow Standard et le framework BMAD v6.

Étapes

1. Initialiser le framework BMAD avec `/workflow:init`
2. Lancer `/workflow:init` "Ajouter un produit au panier"
3. Valider le plan avec `/workflow:plan`
4. Passer la quality gate PRD avec `/gate:validate-prd`
5. Obtenir le design avec `/workflow:design`
6. Utiliser `/sprint:transition` pour passer les stories en in-progress
7. Implémenter avec `/workflow:implement` en suivant le cycle TDD
8. Valider chaque story avec `/gate:validate-story`
9. Utiliser `/sprint:transition` pour passer les stories en review puis done

Critères de succès

- ☐ Track Standard identifié
 - ☐ BMAD initialisé avec `/workflow:init`
 - ☐ PRD généré avec user stories
 - ☐ Quality gate PRD validée (≥80%)
 - ☐ Design technique validé
 - ☐ Stories transitionnées via `/sprint:transition`
 - ☐ Tests écrits en premier (TDD Red-Green-Refactor)
 - ☐ Code implémenté et testé
 - ☐ Story DoD validée (100%)
-

Points Clés à Retenir

1. **3 tracks** : Quick Flow (< 5 min), Standard (< 15 min), Enterprise (< 30 min)
 2. **Analyse obligatoire** : Comprendre → Analyser → Documenter → Valider
 3. **BMAD v6** : Framework de gestion de projet avec quality gates et status routing
 4. **Quality Gates** : PRD (≥80%), Tech Spec (≥90%), Backlog (INVEST 6/6), Sprint Ready (100%), Story DoD (100%)
 5. **Status Routing** : backlog → ready-for-dev → in-progress → review → done (+ blocked)
 6. **TDD** : Test d'abord, implémentation ensuite, refactoring toujours
 7. **Impact** : Évaluer avant de coder, valider si impact moyen/fort
-

Durée estimée : 2h **Prochain module** : Démarrer un Nouveau Projet Symfony