# **Documentation Gestion de Projet**

## **Projet Cinephoria CDA**

**Étudiant :** Romain Rémusat  
 **Formation :** Concepteur Développeur d'Applications  
 **Date :** juillet 2025

## **Méthode de gestion choisie**

### **Kanban - Justification du choix**

**Méthode retenue :** Kanban via Trello

**Raisons du choix :**

**Flexibilité adaptée au projet individuel :** Le Kanban permet d'ajuster les priorités en temps réel selon les difficultés techniques rencontrées ou les évolutions du projet, contrairement à des méthodes plus rigides comme le cycle en V.

**Visualisation claire de l'avancement :** Le tableau permet une vue d'ensemble immédiate de l'état d'avancement, essentiel pour un projet de 90h avec de multiples livrables à coordonner.

**Simplicité de mise en œuvre :** Pour un projet individuel CDA, Kanban évite la complexité administrative de méthodes comme Scrum (pas de daily, sprint reviews, etc.) tout en conservant les bénéfices de l'agilité.

**Limitation du work-in-progress :** Le Kanban impose naturellement de limiter les tâches en cours, évitant la dispersion sur les 3 applications à développer simultanément.

## **Organisation Trello**

### **Structure du tableau Kanban**

**Lien Trello :** https://trello.com/b/dLrilC0o/cinephoria-tp-cda

**6 colonnes organisées selon le workflow :**

#### **1. Backlog (Fonctionnalités prévues)**

* **Objectif :** Recenser toutes les fonctionnalités par ordre de priorité
* **Organisation :** User Stories triées par criticité métier
* **Contenu :** 15 User Stories + tâches techniques (50 cartes total)
* **Priorisation :** MoSCoW (Must have / Should have / Could have)

#### **2. Sprint Actuel (Développement prévu)**

* **Objectif :** Fonctionnalités planifiées pour le sprint en cours
* **Durée :** Sprints de 1-2 semaines selon la complexité
* **Critères d'entrée :** Tâches analysées et estimées
* **Limite WIP :** Maximum 5 cartes pour éviter la dispersion

#### **3. En Cours (Développement actuel)**

* **Objectif :** Tâches en développement actif
* **Limite WIP :** Maximum 2 cartes (principe Kanban)
* **Assignation :** Une seule personne par carte
* **Durée max :** 3 jours pour éviter les blocages

#### **4. Tests (Validation)**

* **Objectif :** Fonctionnalités en cours de validation
* **Types de tests :** Fonctionnels, sécurité, intégration
* **Critères de sortie :** Tests passés + revue de code
* **Documentation :** Résultats de tests documentés

#### **5. Terminé – Develop (Branche dev)**

* **Objectif :** Fonctionnalités validées sur branche develop
* **Statut :** Code testé et fonctionnel
* **Action suivante :** Attente de merge vers main
* **Traçabilité :** Lien vers commit Git

#### **6. Terminé – Main (Production)**

* **Objectif :** Fonctionnalités mergées en branche principale
* **Statut :** Code en production ou prêt pour déploiement
* **Validation finale :** Tests d'intégration réussis
* **Traçabilité :** Tag de version Git

### **Système de label**

**Couleurs et significations :**

* **🔴 Priorité Haute :** Fonctionnalités critiques (US 4, 6-7, 13-14)
* **🟡 Priorité Moyenne :** Fonctionnalités importantes (US 1-3, 8-10)
* **🟢 Priorité Basse :** Fonctionnalités optionnelles (US 15, améliorations)
* **🔵 Frontend :** Tâches interface utilisateur
* **🟣 Backend :** Tâches côté serveur
* **🟠 Mobile :** Développement application mobile
* **⚫ Infrastructure :** Configuration, déploiement, base de données

## **Gestion des branches Git**

### **Stratégie de branching**

**Modèle adopté :** Git Flow simplifié

**Lien GitHub :** https://github.com/RomainRemusat/cinephoria

#### **Architecture des branches :**

main (production)

├── develop (intégration)

├── feature/auth (authentification)

├── feature/reservation (système réservation)

├── feature/admin (interface admin)

├── feature/mobile (app mobile)

└── feature/bureautique (app desktop)

#### **Workflow de développement :**

**1. Création de fonctionnalité :**

git checkout develop

git pull origin develop

git checkout -b feature/nom-fonctionnalite

**2. Développement :**

* Commits atomiques et descriptifs
* Tests unitaires avant commit
* Respect des conventions de nommage

**3. Intégration :**

git checkout develop

git merge feature/nom-fonctionnalite

git push origin develop

**4. Mise en production :**

git checkout main

git merge develop

git tag v1.0.0

git push origin main --tags

### **Conventions adoptées**

#### **Messages de commit :**

feat(auth): ajout système connexion utilisateur

fix(reservation): correction calcul prix total

docs(readme): mise à jour instructions installation

#### **Libellé des branches :**

* **feature/** : nouvelles fonctionnalités
* **bugfix/** : corrections de bugs
* **hotfix/** : corrections urgentes production
* **docs/** : documentation

## **Planification et suivi**

### **Estimation initiale**

**Charge totale estimée :** 90 heures (conforme énoncé)

**Répartition par phase :**

* **Conception :** 15h (17%) - Wireframes, charte, BDD
* **Développement Web :** 45h (50%) - Application principale
* **Développement Mobile :** 15h (17%) - App Flutter/responsive
* **Développement Bureautique :** 10h (11%) - App Python
* **Documentation :** 5h (5%) - Finalisation livrables

**Échéances intermédiaires :**

* **Semaine 1-2 :** Conception et wireframes
* **Semaine 3-6 :** Développement application web
* **Semaine 7 :** Application mobile
* **Semaine 8 :** Application bureautique
* **Semaine 9 :** Tests et documentation
* **Semaine 10 :** Finalisation et rendu

### **Métriques de suivi**

#### **Indicateurs Kanban :**

* **Lead Time :** Temps moyen entre création et livraison d'une carte
* **Cycle Time :** Temps moyen en développement actif
* **Throughput :** Nombre de cartes livrées par semaine
* **WIP :** Respect des limites de travail en cours

#### **Indicateurs Git :**

* **Fréquence de commit :** Objectif 1 commit/jour minimum
* **Couverture de tests :** Objectif 70% code métier
* **Révisions de code :** Relecture avant merge vers develop

#### **Indicateurs qualité :**

* **Bugs détectés :** Traçabilité et résolution
* **Respect planning :** Écarts prévus/réalisés
* **User Stories complètes :** Critères d'acceptation validés

## **Adaptations et ajustements**

### **Défis rencontrés**

**Challenge technique : Transaction SQL atomique**

* **Problème :** Complexité de la gestion des réservations simultanées
* **Solution :** Utilisation de FOR UPDATE et procédures stockées
* **Impact planning :** +2h sur développement backend

**Challenge intégration : MongoDB + MySQL**

* **Problème :** Synchronisation des statistiques
* **Solution :** API REST avec endpoints dédiés
* **Impact planning :** +1h sur architecture

### **🔧 Améliorations du processus**

**Évolution des pratiques :**

* **Ajout étiquettes** Trello pour meilleure visibilité
* **Documentation technique** au fur et à mesure
* **Tests automatisés** pour éviter régressions

**Optimisations futures :**

* **CI/CD pipeline** GitHub Actions
* **Environnements** de staging/production
* **Monitoring** applicatif en production

## **Bilan de la gestion projet**

### **Points forts**

**Organisation efficace :** Le Kanban Trello a permis une visibilité constante sur l'avancement et une priorisation claire des développements.

**Flexibilité :** Possibilité d'ajuster les priorités selon les difficultés techniques rencontrées sans remettre en cause l'ensemble du planning.

**Traçabilité :** Lien clair entre cartes Trello et branches Git, facilitant le suivi des fonctionnalités de l'idée à la mise en production.

**Respect du scope :** Toutes les User Stories ont été traitées dans les délais impartis grâce à une bonne estimation initiale.

### **🔧 Axes d'amélioration**

**Estimation initiale :** Certaines tâches techniques (sécurisation, tests) ont nécessité plus de temps que prévu.

**Documentation :** La documentation technique aurait pu être rédigée plus tôt pour éviter l'accumulation en fin de projet.

**Tests :** Les tests auraient pu être systématisés plus tôt dans le processus de développement.

### **Résultats obtenus**

**Livrables :** 100% des livrables requis produits dans les délais **Qualité :** Code fonctionnel avec sécurisation appropriée **Documentation :** Documentation complète et professionnelle **Architecture :** Solution modulaire et évolutive

## **Conclusion**

La méthode Kanban s'est révélée parfaitement adaptée à ce projet CDA, permettant une gestion efficace des multiples livrables tout en conservant la flexibilité nécessaire au développement. L'association Trello + Git a fourni la traçabilité requise pour un projet professionnel.

L'organisation en sprints courts avec limitation du WIP a évité la dispersion sur les 3 applications à développer, garantissant un focus sur les priorités métier.

Cette approche pragmatique démontre la maîtrise des méthodes agiles et leur adaptation aux contraintes d'un projet de formation, tout en respectant les exigences qualité d'un environnement professionnel.