Cours M110

Adopter l'Approche Agile & Outils DevOps

Formation de 100 heures - De la gestion de projet au déploiement

---

> **Objectif du module :** Apprendre à gérer un projet de développement web de bout en bout, de la compréhension du besoin jusqu'au déploiement, en utilisant des méthodes modernes (Agile) et des outils techniques (Jira, Git, SonarQube, GitLab CI).

PARTIE 1 : Les Fondamentaux de la Gestion de Projet

1. Qu'est-ce qu'un projet ?

Un projet est une action temporaire (avec un début et une fin) visant à créer un produit ou service unique.

Triangle de la gestion de projet (Contraintes)

Tout projet est soumis à trois contraintes liées :

- **Coût** (Budget)
- **Délai** (Temps)
- **Qualité** (Périmètre/Fonctionnalités)

> **Exemple :** Si le client veut le site web plus vite (Délai réduit), cela coûtera plus cher (plus de développeurs) ou on devra réduire les fonctionnalités (Qualité).

2. Les Acteurs (Parties Prenantes)

- **MOA (Maîtrise d'Ouvrage) / Client :** Celui qui exprime le besoin et paie.
- **MOE (Maîtrise d'Œuvre) / Équipe Projet :** Celui qui réalise le projet technique.
- **Chef de projet informatique :** Le chef d'orchestre. Il planifie, coordonne l'équipe et communique avec le client.

3. Cycle en V vs Méthodes Agiles

Cycle en V (Méthode Classique)

Approche séquentielle ("Effet Tunnel"). On définit tout au début, on développe, et le client ne voit le résultat qu'à la fin.

> **Inconvénient : Si le besoin change en cours de route, c'est très coûteux de revenir en arrière.**

Méthode Agile

Approche itérative et incrémentale. On découpe le projet en petits morceaux.

> **Le Manifeste Agile (4 valeurs) :**
>
> 1. Les individus et interactions > Processus et outils
> 2. Logiciel fonctionnel > Documentation exhaustive
> 3. Collaboration avec le client > Négociation contractuelle
> 4. Adaptation au changement > Suivi d'un plan

PARTIE 2 : Planifier un Projet

1. Analyser le besoin

Il faut distinguer les besoins explicites (ce que le client dit) des besoins implicites (ce qui est évident mais non dit).

> **Exemple :**
> **Explicite :** "Je veux un site pour réserver des chambres d'hôtel."
> **Implicite :** "Le site doit être sécurisé (HTTPS) et rapide."

2. Découpage et Estimation (WBS)

On ne peut pas estimer un gros projet d'un coup. On le découpe en tâches.

- **WBS (Work Breakdown Structure) :** Décomposition hiérarchique des travaux.
- **Estimation :** Pour chaque tâche, on estime le temps nécessaire (optimiste, pessimiste, probable).

3. Planification (Gantt & PERT)

- **Chemin Critique (CPM) :** C'est la séquence de tâches la plus longue qui détermine la durée minimale du projet. Si une tâche sur ce chemin est en retard, tout le projet est en retard.
- **Diagramme de Gantt :** Outil visuel pour voir le calendrier des tâches (qui fait quoi et quand).

PARTIE 3 : La Méthodologie SCRUM (Le cœur de l'Agile)

Scrum est la méthode agile la plus utilisée. Elle fonctionne par cycles courts appelés **Sprints** (généralement 2 à 4 semaines).

1. Les 3 Rôles Scrum

Rôle	Description
Product Owner (PO)	Représente le client. Il définit les fonctionnalités (User Stories) et les priorise. Il est responsable de la valeur du produit.
Scrum Master	Le coach. Il s'assure que l'équipe respecte la méthode Scrum et lève les obstacles (ex: un serveur en panne).
Équipe de Développement	Les développeurs, testeurs, designers qui réalisent le travail. Il n'y a pas de hiérarchie interne.

2. Les 3 Artéfacts (Documents)

- **Product Backlog :** La liste complète de toutes les fonctionnalités à faire pour le produit (gérée par le PO).
- **Sprint Backlog :** La liste des tâches sélectionnées pour le sprint en cours uniquement.
- **Incrément :** La version du logiciel fonctionnelle et utilisable livrée à la fin du sprint.

3. Les 4 Événements (Cérémonies)

Événement	Description
Sprint Planning	Au début du sprint. On décide quoi faire (Quoi ?) et comment le faire (Comment ?).
Daily Scrum	Réunion de 15 min max, debout. Chacun dit : ce qu'il a fait hier, ce qu'il fait aujourd'hui, et s'il a des problèmes.
Sprint Review	À la fin du sprint. On fait une démo du produit au client pour avoir son feedback.

Sprint Retrospective	Juste après la review. L'équipe discute de sa façon de travailler pour s'améliorer au prochain sprint.

4. Outil : Jira

Jira est le logiciel utilisé pour gérer ces projets Agile.

- **Épic :** Une grosse fonctionnalité (ex: "Gestion des utilisateurs").
- **User Story :** Une fonctionnalité précise (ex: "En tant que client, je veux m'inscrire pour acheter").
- **Task :** Une tâche technique (ex: "Configurer la base de données").
- **Workflow :** Le cycle de vie d'un ticket (À faire → En cours → Terminé).

PARTIE 4 : Qualité du Code et Versioning

1. Gestion de Versions (Git & GitLab)

Git permet de sauvegarder l'historique du code et de travailler à plusieurs sans s'écraser les fichiers.

Concepts clés

- **Dépôt (Repository) :** Là où le code est stocké.

Commandes essentielles

Commande	Description
`git clone`	Récupérer le projet sur son PC
`git add`	Préparer les fichiers modifiés
`git commit`	Valider les changements (créer un point de sauvegarde local)
`git push`	Envoyer les changements vers le serveur (GitLab)
`git pull`	Récupérer les changements des autres

> **Branching : On ne travaille jamais sur la branche principale (main ou master) directement. On crée des branches pour chaque fonctionnalité, puis on les fusionne (Merge Request).**

2. Qualité du Code (SonarQube)

SonarQube est un outil qui analyse votre code automatiquement pour détecter :

- **Bugs :** Erreurs qui vont faire planter l'app.
- **Vulnérabilités :** Failles de sécurité.
- **Code Smells (Dette technique) :** Code mal écrit, difficile à maintenir (ex: code dupliqué, fonctions trop complexes).

> **La métrique "MFS" :** Maintenabilité, Fiabilité, Sécurité.

PARTIE 5 : DevOps et CI/CD

1. La philosophie DevOps

DevOps vise à casser le mur entre les développeurs (Dev) et les administrateurs systèmes (Ops).

- **But :** Livrer du logiciel plus vite, plus souvent et de meilleure qualité.
- **Moyens :** Communication, Collaboration et surtout Automatisation.

2. CI/CD (Intégration Continue / Déploiement Continu)

C'est un pipeline (tuyau) automatisé qui se lance à chaque fois que vous envoyez du code (`git push`).

CI (Continuous Integration)

- Le code est compilé automatiquement (Build).
- Les tests sont lancés automatiquement.
- **Intérêt :** On sait tout de suite si on a cassé quelque chose.

CD (Continuous Delivery/Deployment)

- Si la CI est valide (tests OK), le code est déployé automatiquement sur un serveur de test ou de production.

> **GitLab CI :** On configure ce pipeline grâce à un fichier `.gitlab-ci.yml` à la racine du projet. On y définit les **Stages** (étapes : build, test, deploy) et les **Jobs** (tâches).

📝 Résumé Synthétique pour l'examen

Agile/Scrum

C'est travailler par cycles courts (Sprints), livrer souvent, et s'adapter aux changements. **Rôles clés :** PO, Scrum Master, Team.

Jira

L'outil pour suivre les tâches du Sprint (Tableau Kanban : To Do, In Progress, Done).

Git

Indispensable pour ne pas perdre son code et fusionner le travail de l'équipe.

SonarQube

Le "correcteur orthographique" de votre code (qualité et sécurité).

DevOps (CI/CD)

Automatiser tout ce qui est répétitif (tests, mise en ligne) pour aller vite sans faire d'erreurs manuelles.

Cours M110 - Approche Agile & Outils DevOps - 100 heures