

**PROJET** = TACHES + DUREE + ORDONNANCEMENT + OBJECTIF

**RESSOURCES** = LES ELEMENTS NECESSAIRES POUR REALISER LE PROJET

**GESTION DE PROJET** = ORGANISATION + CONDUITE

**LIVRABLE** = RESULTAT FINALE DU PROJET

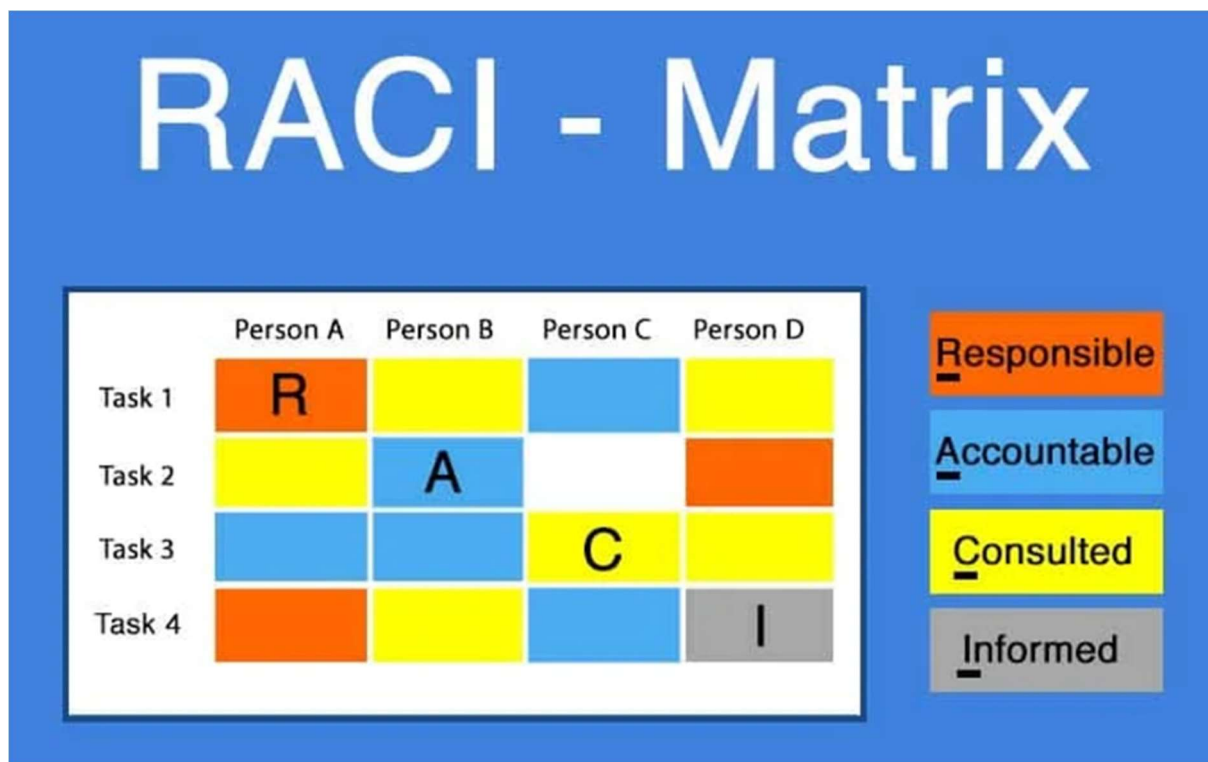
**Une charte de projet** = RAPPORT DE PROJET

**Parties prenantes** = ACTEURS EXTERNES ET INTERNES

**GESTION DES RISQUES** : MATRICE DES RISQUES

|    | RISQUE                                      | GRAVITÉ | PROBABILITÉ DE RISQUE | SOLUTION   |
|----|---|---------|-----------------------|--|
| R1 | Financier: Manque de budget                 | LEGER   | TRES PROBABLE         | la collaboration financière des membres d'équipe   |
| R2 | Humains : démission au cours du projet      | MAJEUR  | POSSIBLE              | on va déviser les tâches   |
| R3 | Temporels : mauvaise estimation des délais, | SEVERE  | TRES PROBABLE         | doubler les effort pour réctifier  |
| R4 | Matériels: pannes                           | SEVERE  | POSSIBLE              | Achat d'un nouveau matériel si possible sinon travailler avec un matériel de l'établissement |
| R5 | Humains: conflits au sein de l'équipe,      | MAJEUR  | TRES PROBABLE         | Gérer ce conflit d'une manière logique et approprié  |
| R6 | Humains: manque de compétences              | MAJEUR  | TRES PROBABLE         | Programmer des réunion pour consulter la progression   |

## **RACI MATRIX**



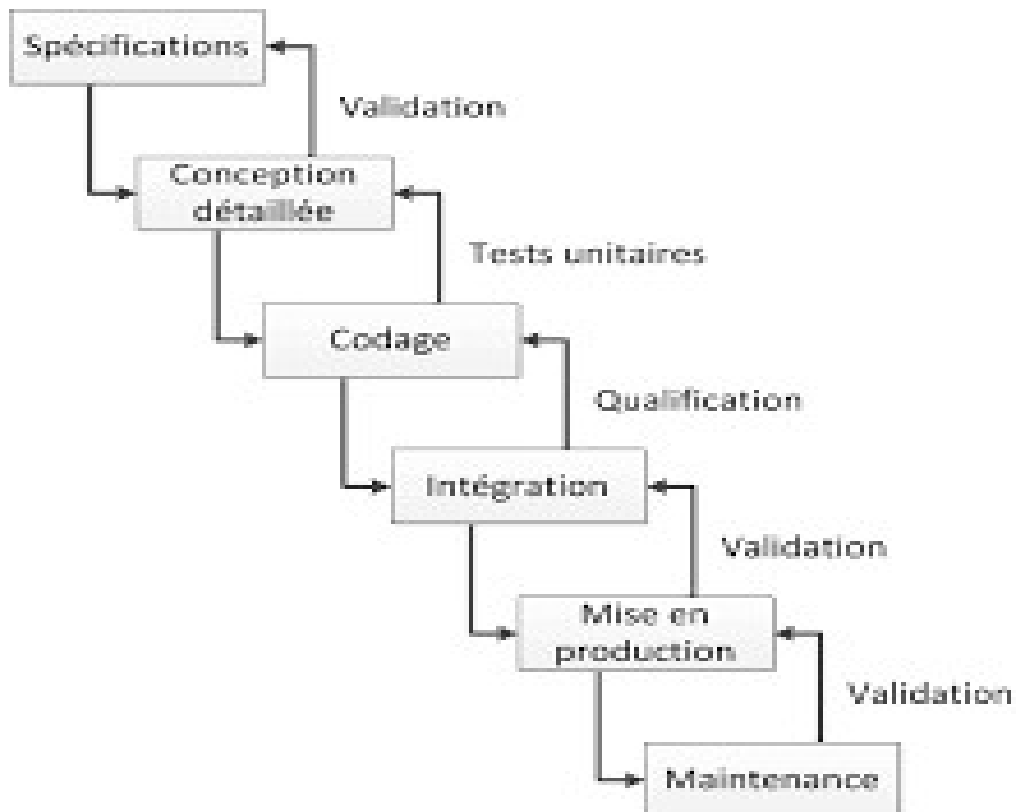
## **MATRICE DES COMPETENCES**

## MATRICE DES COMPETENCES

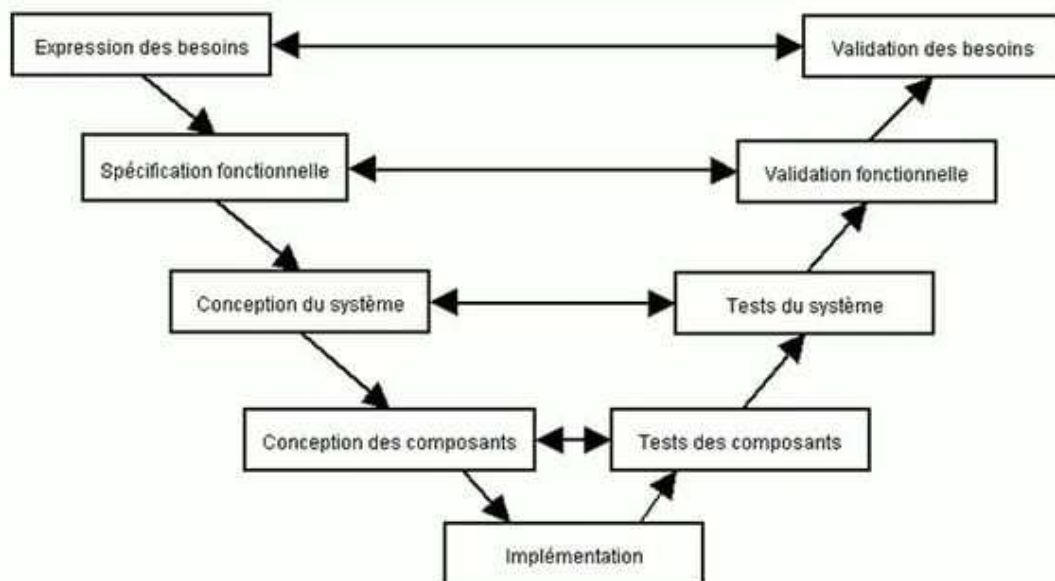
| Nom/<br>Compétences      | Attendu   | Yassine | Naima | Soufiane | Oussama |
|--------------------------|-----------|---------|-------|----------|---------|
| HTML/CSS                 | ● ● ● ●   | ●       | ●     | ●        | ●       |
| JAVASCRIPT               | ● ● ● ● ● | ●       | ●     | ●        | ●       |
| REACTJS                  | ● ● ● ● ● | ●       | ●     | ●        | ●       |
| (Laravel/<br>expresseJS) | ● ● ● ● ● | ●       | ●     | ●        | ●       |
| (Mysql/<br>Mongodb)      | ● ● ● ● ● | ●       | ●     | ●        | ●       |
| Git                      | ● ● ● ● ● | ●       | ●     | ●        | ●       |

**Rôles DANS UN PROJET** = MISSIONS (chef de projet , MOA , scrum master ...)

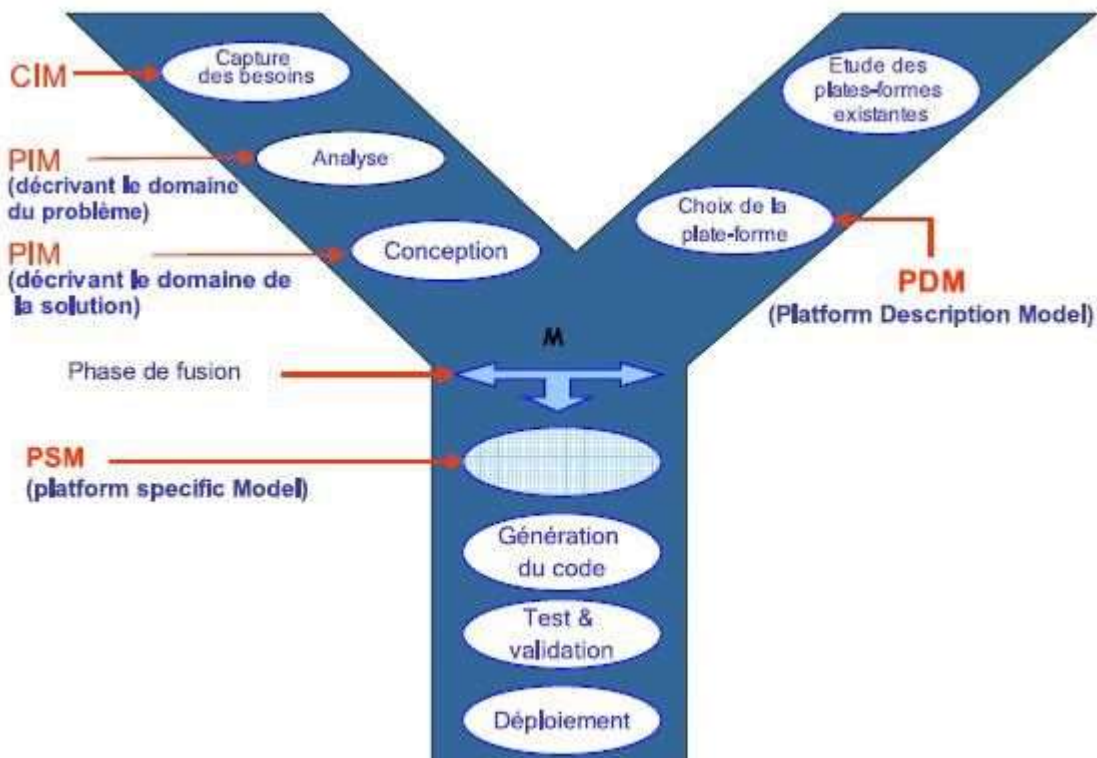
**LES METHODES : CASCADE , Y , V**



## LES METHODES :V



## LES METHODES : Y ou 2TUP



## DECOUPAGE DES FONCTIONNALITES EN TACHES

**Approche descendante** : on part du résultat global et on décompose le projet en sous-projets de plus en plus détaillés, puis en groupes de tâches.

**Approche ascendante** : On note d'abord toutes les tâches qui nous viennent à l'esprit, on les regroupe en groupes de tâches, puis on affecte ces derniers à des sous-projets.

**Approche combinée** : C'est une combinaison des deux techniques précédentes. On liste les tâches. On note ensuite les sous-projets et on leur attribue les tâches précédemment listées.

## BESION =

**BESION EXPLICITE** = CLAIRE (le client possède un cahier des charges détaillées )

**ET BESION IMPLICITE** : (NON CLAIRE)

## COMMENT CAPTURER LE BESION CLIENT =

- INTERVIEW DIRECTE (bureau)
- MAIL à distance
- APPEL à distance
- REUNION EN LIGNE à distance
- Brainstorming
- Qqocqcp
- Questionnaire Google formes
- Enquête
- Groupes dans les réseaux sociaux

Contexte du projet = environnement + organisation

Le périmètre du projet : **scope** du projet = les limites du projet

Par exemple projet RDV médecin est limité sur les dentistes

## Estimation de la durée de réalisation de chaque tâche

⌘ Données **historiques** (ou estimation analogue)

⌘ Analyse statistique et mathématique (ou estimation paramétrique)




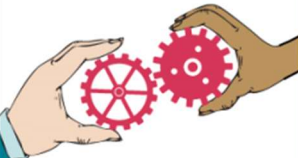



**jour-homme** correspond au travail d'une personne pendant un jour. Un projet présentant 10 jours-homme de travail peut être réalisé en 10 jours par 1 personne, en 1 jour par 10 personnes, en 20 jours par une personne disponible à 50%,

LES METHODES AGILES : SCRUM, KANBAN, SCRUMBAN, LEAN

## 12 PRICIPES DE L'AGIL

Knowledge  
TRAIN

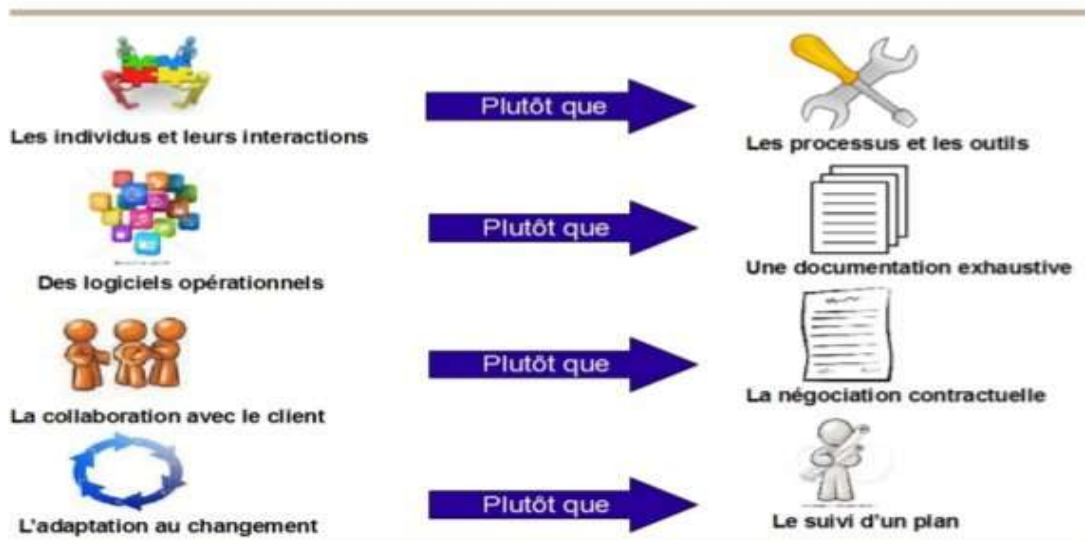
# Les 12 principes Agile \*

|  |   |   |  |
|--|---|---|--|
| <b>1 Satisfaire le client</b><br> | <b>2 Accueillir les changements</b><br>  | <b>3 Livrer fréquemment</b><br>       | <b>4 Travailler ensemble</b><br>      |
| <b>5 Confiance et soutien</b><br> | <b>6 Dialogue en face à face</b><br>     | <b>7 Logiciel opérationnel</b><br>    | <b>8 Développement soutenable</b><br> |
| <b>9 Attention continue</b><br>  | <b>10 Privilégier la simplicité</b><br> | <b>11 Équipes auto-organisées</b><br> | <b>12 Réfléchir et adapter</b><br>   |

Copyright © 2016 Knowledge Train Limited. www.knowledgetrain.co.uk. \*Quoted from www.agilemanifesto.org

## MANIFEST DE L'AGILE

## Le manifeste agile (1.5mn)



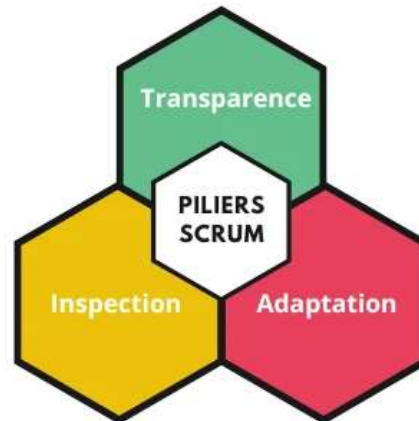


## PILIERES DE L'AGIL

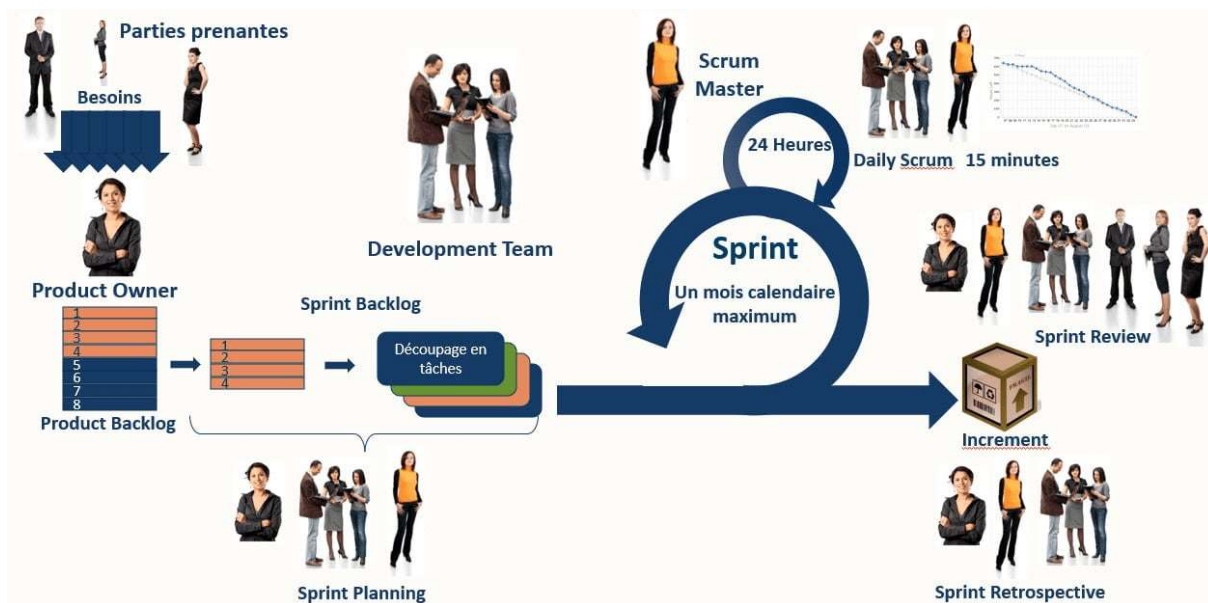


### PILIERES SCRUM

- 01 Nécessite une relation de confiance et une vision commune
- 02 Se fait durant tout le cycle Scrum pour détecter les écarts : Objectifs/réalité
- 03 Exige de s'adapter en fonction des résultats d'une inspection régulière



## LES ARTEFACTES DE SCRUM = LES COMPOSANATS DU SCRUM



## USER STORIES :

Ajouter un produit au panier

- **En tant que :** Utilisateur du site e-commerce
- **Je veux :** Pouvoir ajouter un produit au panier lorsque je trouve un article qui m'intéresse
- **Afin de :** Pouvoir effectuer un achat ultérieurement en regroupant tous les articles que je souhaite acheter.
- **Critères d'acceptation :** En tant qu'utilisateur, je devrais pouvoir cliquer sur un bouton ou une icône clairement identifiable à côté de chaque produit affiché sur la page du produit.
- Lorsque je clique sur ce bouton, le produit doit être ajouté à mon panier sans recharger la page.

**PRODUCT BACKLOG** = TABLEAU des fonctionnalités

**PLANING MEETING** : PLANIFICATION DES MEETING (REUNIONS)

**Le Sprint Backlog** **SOUS TABLEAU** , est une partie sélectionnée du Product Backlog pour un sprint spécifique.

**Product owner** : le client

**SCRUM MASTER** : assure que les principes et les valeurs du Scrum sont respectés.

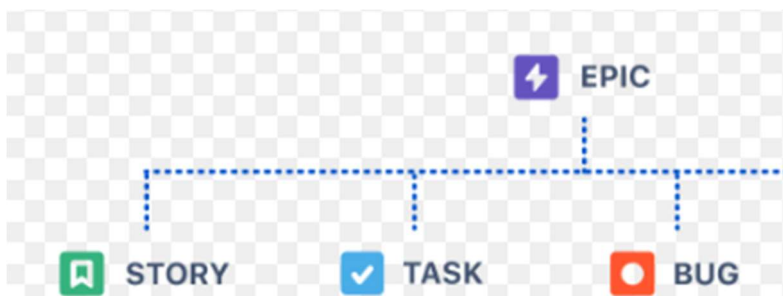
**STAND UP DAILY MEETING** = REUNION JOURNALIERS ENTRE LES DEVELOPPEURS ET CHEF DE Projet et SCRUM MASTER

## SPRINT REVIEW VS SPRINT RETROSPECTIVE :

|               | SPRINT REVIEW                       | SPRINT RETROSPECTIVE :                 |
|---------------|-------------------------------------|--|
| PLANIFICATION | LA FIN DU SPRINT                    | APRES SPRINT REVIEW,                   |
| MEMEBRES      | ACTEURS INTERNES ET <b>EXTERNES</b> | ACTEURS INTERNES                       |
| SUJET         | REMARQUES, LES BUGS                 | AMELIORATION POUR EVITER LES PROBLEMES |

## EPIC

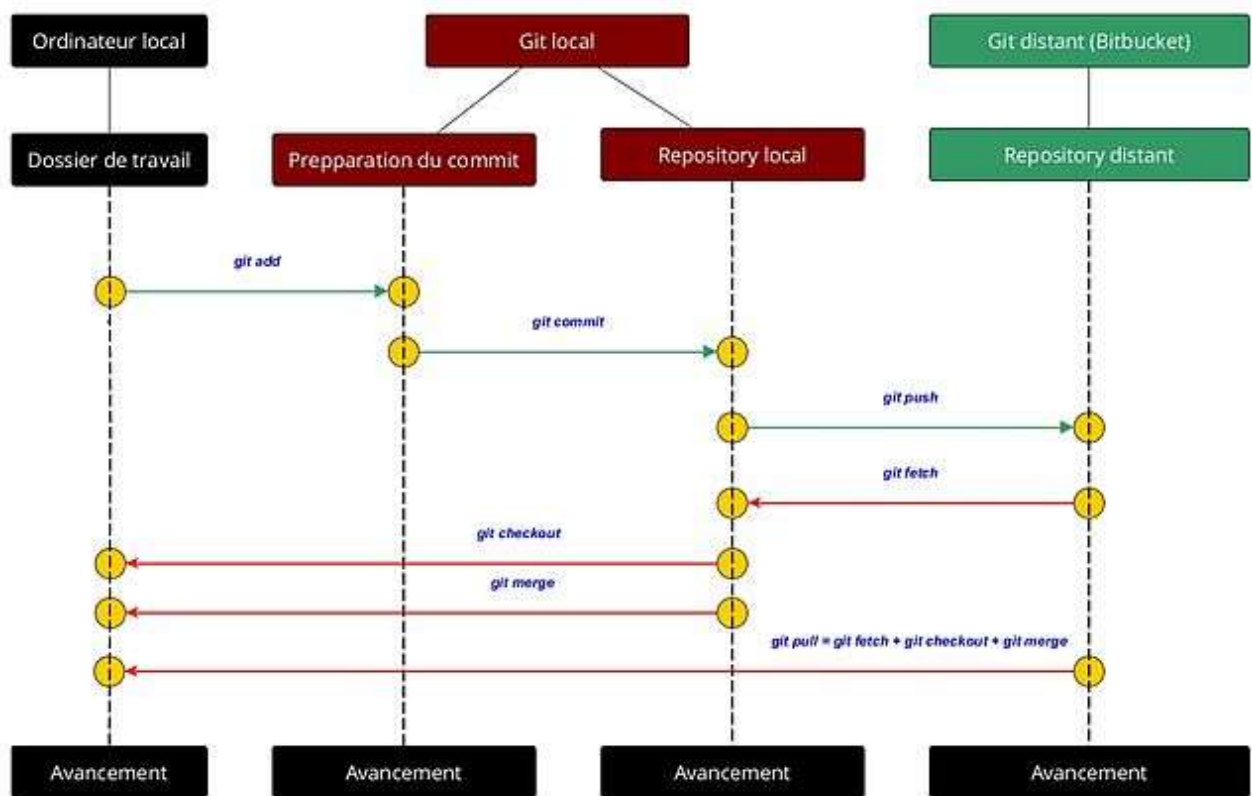
un regroupement de plusieurs fonctionnalités du système à développer



**Processus de la méthode Scrum** : pre-game, game, postgame

1. **Pre-Game**(planification + architecture)
2. **Game**(Sprint + réunion Scrum)
3. **Post-Game**(démonstration + clôture)

## Git hub / Gitlab



Les principales commandes git sont :

- git **add** : **ajoute** des fichiers depuis le **répertoire de travail** vers **staging area** .
- git **commit** : **ajoute** des fichiers depuis **staging area (index)** vers **comitted files(head)** .
- git **push** : **publie** des modifications depuis **comitted area** vers **remote repository**.
- git **fetch** : **téléchargera** le contenu de **remote repository**, ne changera pas l'état du dépôt local.
- Git **checkout** : **basculer** d'une branche à l'autre
- Git **merge** : permet de **fusionner** différentes branches créées
- git **pull** : **recupère** des modifications depuis **remote repository** vers **comitted files(head)** , changera l'état du dépôt local.
- git **clone** : permet de créer une **copie** ou un de dépôts distants
- git **init** : **crée** un nouveau dépôt vide à l'emplacement courant.
- git **status** : **affiche** les différences entre le répertoire de travail, l'index et HEAD.

## Sonar Qube :Les métriques

➤ SonarQube génère un rapport consultable via un navigateur sur :

- Densité des **commentaires**
- Taux de couverture des **tests** unitaires
- des conventions de **nommage**
- Respect des règles de codage et des bonnes pratiques
- Détection de **bogues**
- Détection de **code mort**
- Détection de code **dupliqué**
- **Complexité** du code (complexité cyclomatique, complexité cognitive)
- Score de **maintenabilité**,



- Fiabilité et **sécurité**
- Dette technique (estimation du **temps nécessaire pour fixer tous les problèmes** détectés )

### Sonar couvre les 7 axes de la qualité du code :

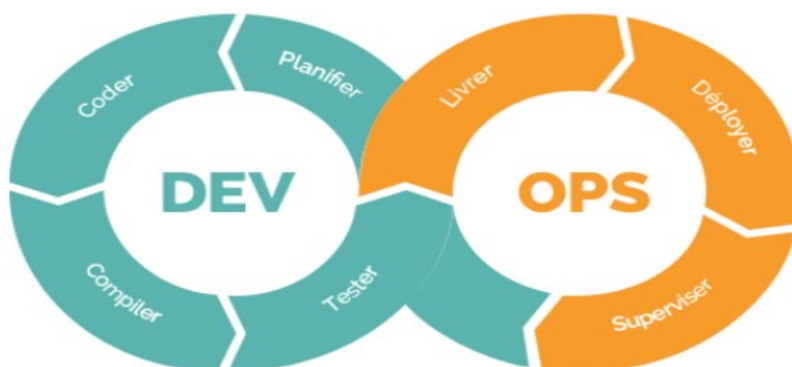
1. architecture & design ,
2. documentation ,
3. respect des standards de codage ,
4. non duplication du code ,
5. tests unitaires ,
6. complexité ( rapidité de l'algorithme )
7. bogues potentiels.

| <div> <div>☆ Aliquot</div> <div>Quality Gate: <span>Passed</span></div> </div> <div> <div> <div>A</div> <div>Reliability</div> </div> <div> <div>A</div> <div>Security</div> </div> <div> <div>A</div> <div>Maintainability</div> </div> <div> <div>0.0%</div> <div>Couverture (TU)</div> </div> <div> <div>0.0%</div> <div>Duplications</div> </div> <div> <div>XS 784</div> <div>PHP, JavaScript</div> </div> </div> |                         |                                       |  |
|--|-------------------------|---------------------------------------|--|
| Score  | Fiabilité               | Sécurité                              | Maintenabilité                                     |
| <b>A</b>   | 0 bug                   | 0 vulnérabilité                       | $0,00 \leq \text{ratio dette technique} \leq 0,05$ |
| <b>B</b>   | Au moins 1 bug mineur   | Au moins 1 vulnérabilité mineure      | $0,06 \leq \text{ratio dette technique} \leq 0,1$  |
| <b>C</b>   | Au moins 1 bug majeur   | Au moins 1 vulnérabilité majeure      | $0,11 \leq \text{ratio dette technique} \leq 0,20$ |
| <b>D</b>   | Au moins 1 bug critique | Au moins 1 vulnérabilité minecritique | $0,21 \leq \text{ratio dette technique} \leq 0,50$ |
| <b>E</b>   | Au moins 1 bug bloquant | Au moins 1 vulnérabilité bloquante    | $0,51 \leq \text{ratio dette technique} \leq 1$    |

### Dev ops

Dev ops = développement + opération (administrateur réseau )

Dev ops = **intermédiaire**, assure **automatisation** des taches et **optimisation** ( temps et le cout , qualité )



## Les avantages devops

- Collaboration
- Vitesse
- L'agilité
- La satisfaction du client
- L'innovation
- La sécurité DevSecOps

## Les piliers de la structure DevOps

