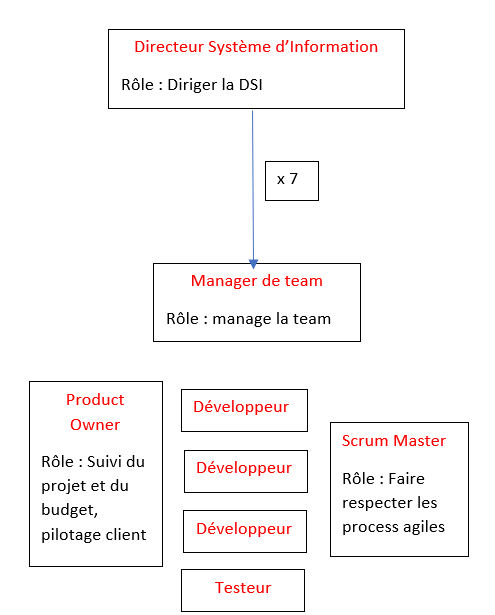
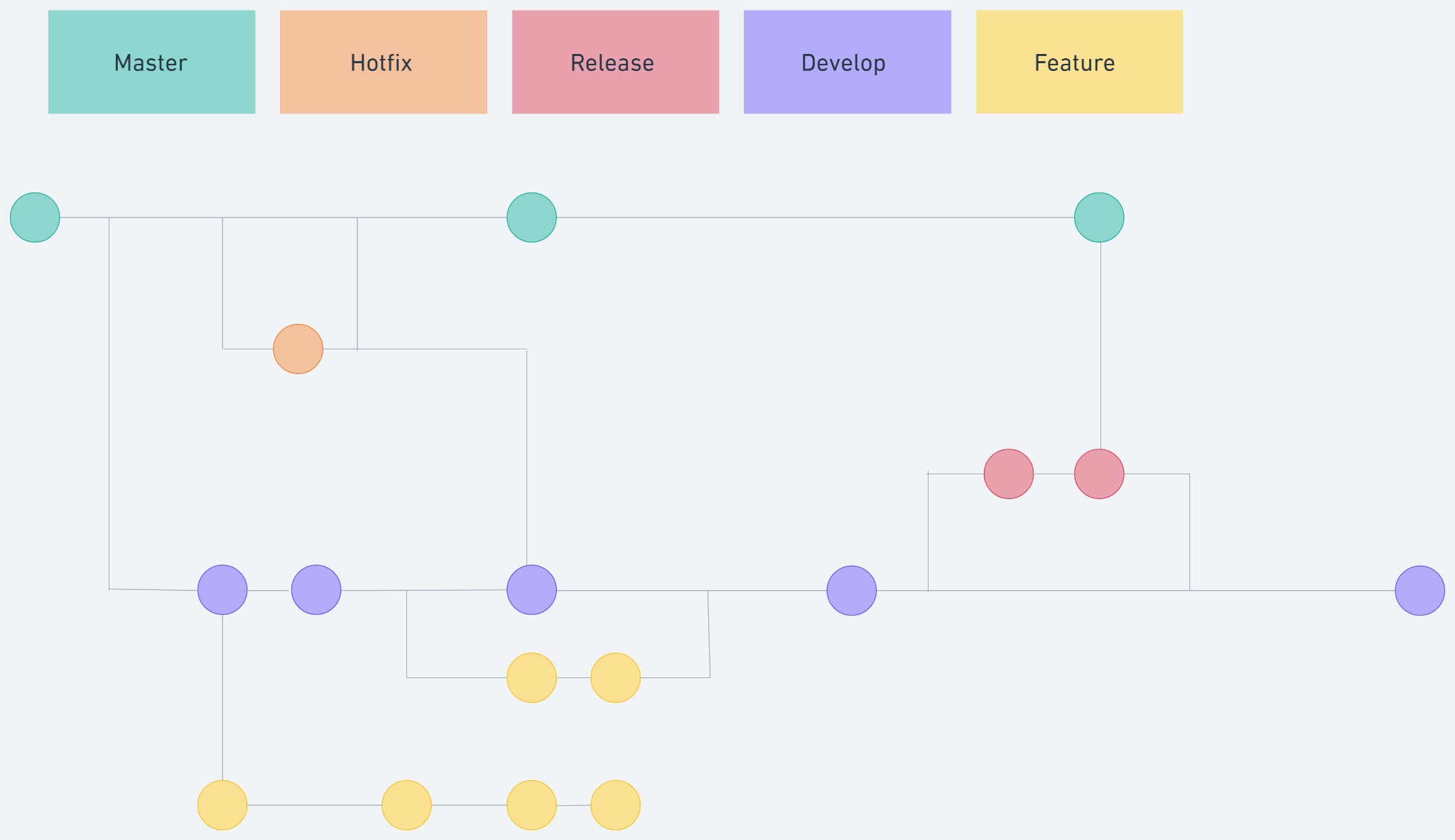
“Mindset” à Gilles

**Un organigramme complet de la DSI avec les rôles de chacun (Vous avez le droit de donner les rôles comme vous le souhaitez tant qu’un suivi formation est donné lors de la présentation).**

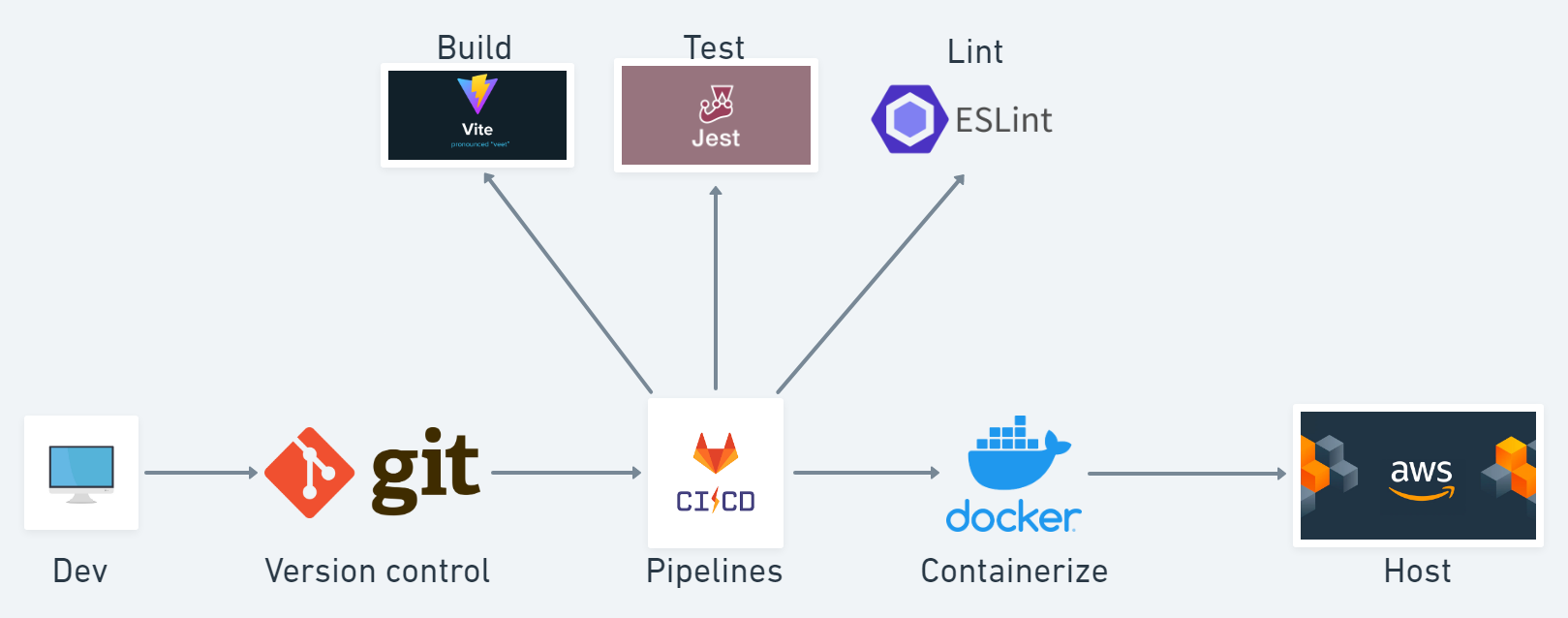


**Un schéma de la gestion des branches sur le gestionnaire de code source**

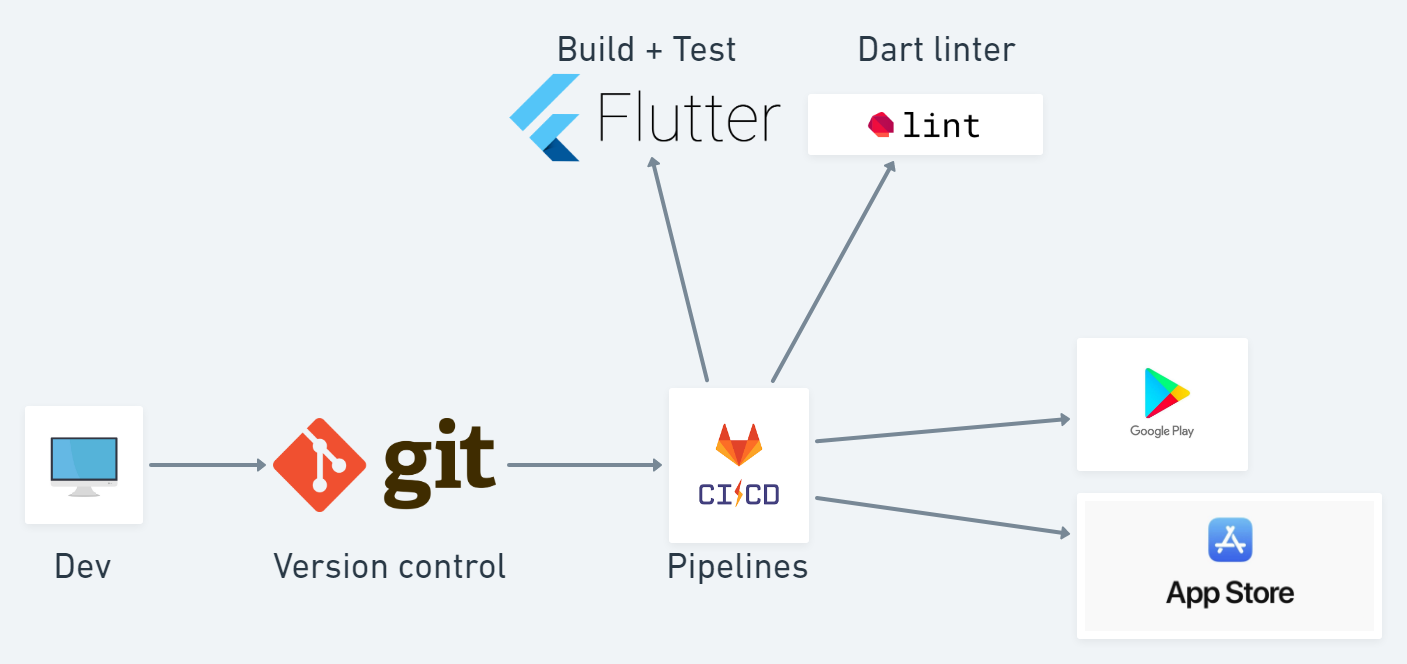


**Un schéma technique expliquant comment l’intégration continue fonctionne (Outils, Cloud, Docker, Kubernetes)**

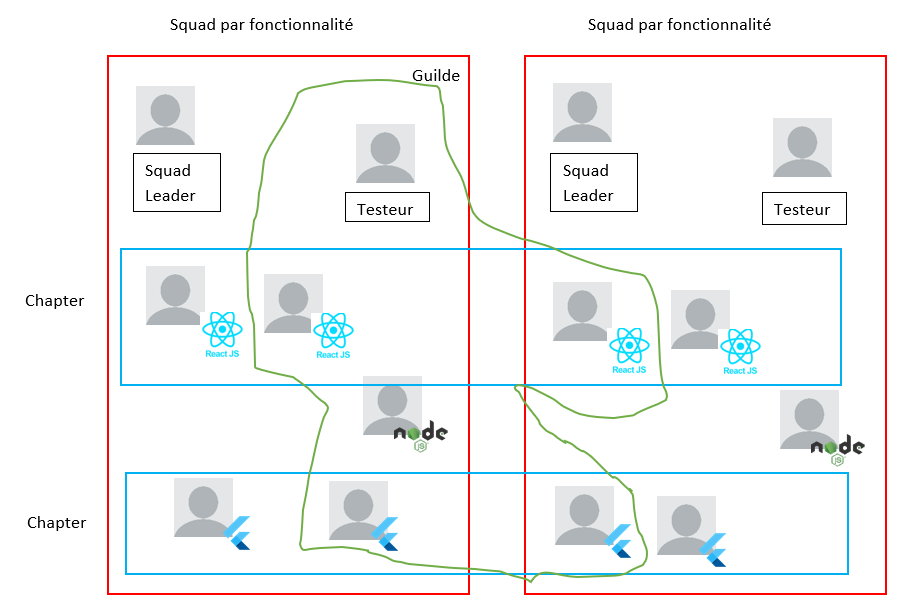
CI pour l’application web Reactjs



CI pour l’application mobile Flutter



**Un schéma sur l’organisation de la DSI en mode Produit (Scrum, Kanban, UX/UI, Modèle spotify)**



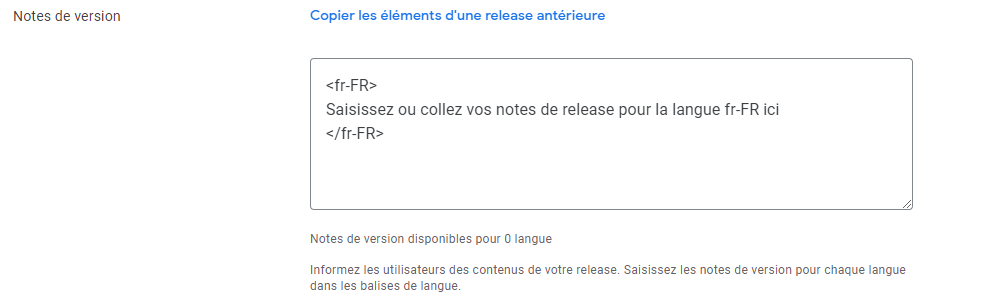
**Un schéma sur le release management à savoir à quelles fréquences sortent les releases en production (Exemple: Toutes les semaines, quelle est la communication donnée aux utilisateurs de l’application par quel moyen sont-ils informés ?) Exemple de release note.**

Rythme des releases:

Pour l’app Android/iOS le rythme entre deux releases est de minimum 3 semaines sans compter les patchs et correctifs.

Pour le site le rythme suivra celui des sprints sans prendre en compte des correctifs/patchs.

Pour l’app android/iOS, mise en place d’une AlertDialog informant l’utilisateur des nouveautés de la release + rajouter dans le playstore dans la partie “note de version” les informations de la release.



Pour le site web, on met en place un système de carrousel informant l’utilisateur des nouveautés lors du clique sur le carrousel contenant l’image d’annonce.

**Une explication de la stratégie de test à mettre en place dans un mode agile avec les rôles qui vont bien ainsi que l’écriture de ces derniers (ex: TDD, BDD, Gherkin).**

La personne chargée de cette stratégie sera le testeur.(il sera dans le coin d’une pièce car personne ne l’aime)

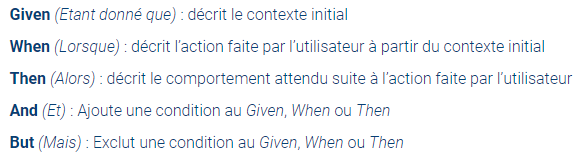
La stratégie que nous utiliserons sera le gherkin. Cette stratégie permet de faciliter la compréhension des tests, de plus le turn-over étant souvent présent dans les équipes informatiques il faut permettre un transfert facile des cas de test, le gherkin étant un langages quasi-textuel sa compréhension est simple.

Le PO pour écrire ses US (user story)

Le QA agile pour écrire ses critères d’acceptation et ses scénarios de test

Le Développeur pour scripter les tests automatisés

Ce à quoi le ressemble le



**Donner un template référence d’une bonne user story (Description de l’user story, critère d’acceptation, exemple de cas de test)**

-Qualifier le besoin

Création du story mapping par le PO avec l’équipe

-Formalisation de la user story

Pouvoir comprendre directement qui exprime le besoin, quel est le besoin auquel il va falloir répondre et quelle est la finalité

-Vérifier la qualité de la user story

Utilisation de la grille INVEST afin de vérifier qu’elle est indépendante, négociable, valorisable, estimable, small et testable

-Estimer les user stories

Vérifier qu’elle est réalisable en un sprint sinon la redécouper, inciter l’équipe à se projeter de manière concrète, gérer la planification du sprint en priorisant et donner de la visibilité sur ce que l’équipe s’engage à réaliser en un sprint (Utilisation de la suite de Fibonacci / planning poker / bucket system / T Shirt size)

-Déterminer les critères d’acceptance

Rédaction d’un scénario avec Gherkin (Mode d’emploi vérifiant que la solution correspond bien au besoin initial)

-Qualifier l’état de la user story

-Le DoD garantit que tout le monde dans l’équipe (PO compris) sait exactement ce qui est attendu. Il assure la transparence et la qualité du produit.

-La Definition of Ready (DOR) est la liste d’éléments attendus qu’une user story doit rassembler pour être candidate au développement. Ce sont les critères qui vont permettre à la user story de passer d’un état à un autre : du backlog produit au backlog sprint par exemple. La DOR conditionne ainsi le passage en développement de la user story.

**Mettre en place des indicateurs de suivi de la mise en place du DevOps (Ex: Couverture de test, Dette technique, taux de retour de bug après livraison etc..)**

La couverture de test devra couvrir au minimum 80% du code afin de valider le push.

Le taux de fuite de bugs doit être maintenu à un seuil inférieur à 10%, il correspond au ratio entre le nombre de bugs trouvés en production et celui détecté en test.

Mise en place d’une fréquence de déploiement(sprint toutes les deux semaines).

Mise en place d’analyse du “Temps moyen de détection des erreurs (MTTD)” et “Délai moyen de correction (MTTR)”

patch - avantage et inconvénient

**Une documentation expliquant les bonnes pratiques de code à respecter pour garder un code maintenable. (Ex: Linter, design pattern, documentation, Definition of Done). Cette documentation doit être rendu sous forme d’un README markdown git**