IDV-MOB5 / Mon Agenda Partagé / Étape 2 : Stratégie de tests

Pour ce projet nous suivrons le concept du Test Driven Development en concevant les tests en amont en accord avec la logique business de notre application afin de simplifier sa conception et accélérer sa mise en production.

Dans un premier temps, nous mettrons en place les tests unitaires avec jest en javascript pour notre application react native ce qui nous permettra d'adapter le développement aux tests que nous aurons écrit en amont.
Les tests unitaires seront initialement conçus pour guider les développeurs et de mettre en avant les problèmes de logique ou de conception les plus évidents et de les éviter.
Les tests unitaires échoueront au début du développement puis seront validés au fur et à mesure du développement du produit, selon un cycle "red-green-refactor cycle".
Les tests unitaires sont une garantie d'un fonctionnement purement technique mais sont indépendants de la logique induite par le cahier des charges. Ils testent de très petits blocs de code.

Les tests unitaires seront déployés sur un pipeline de test gitlab dans une logique d'intégration continue afin de ne valider uniquement les commit qui respectent les spécifications des tests unitaires.

Une fois ces spécifications respectées et que plus aucun test unitaire n'est refusé, nous entamerons la rédaction des tests d'intégration afin de vérifier sur une hauteur légèrement plus élevée que nos composants ou services interagissent correctement les uns avec les autres.
On peut diviser les tests d'intégration en deux parties:

- Les tests d'intégration composants pour vérifier les unités de code interagissent bien ensemble dans le projet (proche du test unitaire)
- Les tests d'intégration système vérifient quant à eux la bonne interaction et le bon fonctionnement entre ces mêmes unités de code et des composants extérieurs comme par exemple la base firebase que nous allons utiliser ou l'appel à n'importe quelle api si le besoin s'en fait sentir.

Nous mettrons ensuite en place les tests fonctionnels là encore avec jest pour simuler un parcours utilisateur et vérifier que la logique globale de l'application répond bien aux attentes du client.

Si nous estimons que le comportement métier est couvert par les tests fonctionnels nous ne rédigeons pas de tests d'acceptation.

Enfin nous finirons par développer des tests de bout en bout afin de tester l'aspect fonctionnel du système dans son intégralité. Nous pouvons par exemple tester l'intégralité d'un parcours utilisateur dans plusieurs scénarios et vérifier son intégrité.

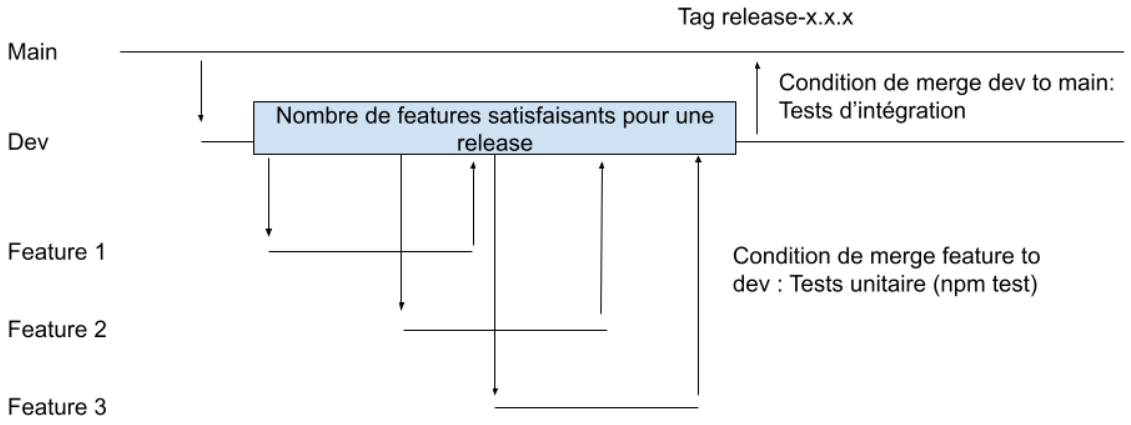

Schéma représentatif du cycle développement-mise en production selon le principe d'intégration continue.