**SOMMAIRE**

DEDICACE

REMERCIEMENTS

INTRODUCTION

1. APPROCHE METHODOLOGIQUE
2. Analyse de l’existant
3. Description de l’application
4. L’architecture
5. Description des fonctionnalités
6. Les limites de l’existant
7. Points forts
8. Points faibles
9. PROPOSITION DE SOLUTION
10. Définition des objectifs
11. Objectifs généraux
12. Objectifs spécifiques
13. Présentation de la solution
14. Niveau technique
15. Niveau organisationnel : architecture

INTRODUCTION

De manière générale, dans l’industrie logiciel la mise en production est non seulement une étape ultime attendue impatiemment par le métier ou le client mais aussi, constitue un véritable parcours de combattant pour toutes les équipes qui interviennent dans la chaine.

En effet, ce processus fait intervenir plusieurs équipes : les développeurs, opérationnels et testeurs. Et généralement les objectifs sont différents biens qu’on final tous sont unanime sur une seule chose celle que l’application soit en production.

Les développeurs cherchent à innover et faire évoluer leur code en mise sur l’application des bonnes pratiques en la matière. Les opérationnels de leur côté cherchent à stabiliser le système d’information. Quant aux testeurs ils se concentrent sur les différents rapports et à documenter les différents tickets ou defects à remonter à l’équipe de dev. En plus de travailler de manière dispersé, chaque équipe utilise ces propres outils. Cette situation aboutit toujours à des situations conflictuelles surtout quand les bugs commencent à apparaitre. Les conséquences qui en découlent sont le retard dans les différentes livrables, des coûts supplémentaires et la satisfaction du client est remise en cause bien que celle-ci est un enjeu de détails. Il devient plus que primordiale d’adopter une autre approche qui permette d’unifier les équipes et les faires parler le même langage tout pour satisfaire le client. Ainsi, développeurs et opérationnels et testeurs travaillent ensemble avec un même processus et des outils connus de tous c’est qu’on appelle le DevOps. Cette notion est basée sur la synergie entre les développeurs et les opérationnels qui est l’alignement de l’ensemble des équipes du système d’information sur un objectif commun tout en réduisant les conflits et éviter le retard dans les livraisons. C’est dans ce cadre que ce positionne notre projet de fin d’études.

Il s’agit pour nous de montrer comment tirer un large profit de l’adoption du DevOps en mettant en place un processus d’intégration continue/livraison contenue avec les meilleurs dans un projet concret. Le projet utilisé est constitué d’une partie front(front-end) et d’une partie back(back-end). La première est conçues avec du Nodejs et la séconde est basé sur l’architecture microservice réalisé avec springboot et java

1. APPROCHE METHODOLOGIQUE
2. Analyse de l’existant

Objectif

Notre objectif est de d’ecrire la solution utilisée avant celle du DevOps, montrer les limites.

1. Description de la solution et les méthodes de travail

L’application est développée dans un environnement AGILE SCRUM. Au sein de l’équipe de développement, il y’a deux sous équipes : une dédiée au front et l’autre au back. Le code se trouve sur un outils de versionning de code (Github). Il y’a deux differents dépots(back et front) te pour chaque dépôt github, il existe 4 branches :

* Devlop(branche dediée à la fusion des dévéloppements)
* Stage( pour les tests réalisés par l’équipe des testeurs)
* Master(dédiée au code final et stable pour la production)
* Release(copie de la master avant chaque nouvelle mise en production)

Chaque dévéloppeur crée sa barnche à partir de la branche devlop. Et dès qu’il finit ses travaux, il pousse son code sur cette branche dont le merge passe par un pull request qui est une étape de validation de son travail. Une fois que son travail est conforme aux règles definis dans l’outils de contrôle de la qualité du code(SonarQube), alors son code est mergé dans dans celui de la branche sans conflit au cas contraire, il est notifié du rejet de sa demande de merge avec des détails pour l’aider à corriger les parties qui ne sont pas alignées avec la politique de qualité.

INTRODUCTION