**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 logicielle, 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 misant 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 soit un enjeu de détails. Il devient plus que primordiale d’adopter une autre approche qui permette d’unifier les équipes et les faire parler le même langage autour d’un objectif commun, celui de la satisfaction du client avec des livrables de qualités et dans les délais impartis. Ainsi, développeurs, 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 au sein d’une équipe en mettant en place un processus d’intégration continue/livraison contenue avec les meilleurs outils en la matière 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 seconde est basée sur l’architecture microservice réalisé avec java et springboot.

1. APPROCHE METHODOLOGIQUE
2. Analyse de l’existant

Objectif

Notre objectif est de décrire le mode de fonctionnement c’est-à-dire les différentes interactions des différentes équipes concernées avant l’adoption de la culture DevOps d’une part, et, d’autre part, 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 dirigée par un scrum master, il y’a deux sous équipes : une dédiée au front-office et l’autre au back-office. La prémière s’occupe de tous ce qui concerne l’expérience utilisateur(interfaces d’utilisation, ergonomie, design…). Il s’ajoute une couche au-dessus de la logique qui est sensée faire les traitements et qui est abstrait pour l’utilisateur. Quant à la seconde, c’est elle qui met en place toute la logique métier définie par les différentes règles de gestions.

Ces deux équipes travaillent en parfaite harmonie car il faut que les fonctionnalités soient techniquement mises en place par une avant que l’autre ne vienne y ajouter une couche pour masquer cette complexité technique via des interfaces astreignantes et cohérentes. Pour se faire, le code source doit être accessible par tous.

Le code se trouve sur un outil de versionning de code (Github). Il y’a deux différents dépôts (back et front) et pour chaque dépôt github, il existe 4 branches :

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

Chaque développeur crée sa branche à partir de la branche develop. 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 définis dans l’outils de contrôle de la qualité du code(SonarQube), alors son code est fusionné (mergé) 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. Une fois les objectifs du sprint atteind, les dev preparent alors une livraison à l’équipe de teste. Cette livraison doit forcément se faire sur un autre environnement qu’on appelle l’environnement de teste ou de qualification afin de permettre aux développeurs de continuer sur les autres sprints. Celui-ci est mis en place par l’équipe des opérationnels ou appelé aussi équipe système.

Du côté des opérationnels, c’est eux qui ont en charge toute l’infrastructure. Ils maintiennent et veillent sur la stabilité des environnements existants, en fournissent de nouveaux en fonction des besoins…

Un environnement est générale une machine physique ou virtuelle sur laquelle est montée toutes les ressources nécessaires pour produire des cas d’utilisation conforment aux règles métier. Généralement, il y’a peu ou quasiment pas de différence avec l’environnement de production. Toutefois, cet environnement n’est pas accessible au client final. Il est y’a un autre environnement qu’on n’appelle pre-prod. C’est ce dernier qui est généralement ouvert aux équipes de teste du client final afin aussi de valider le travailler fournir par toute l’équipe (dev, testeurs…) du prestataire en charge de développer la solution. Quand cette étape est validée, alors l’application est déployé sur le dernier environnement appelé l’environnement de production.

INTRODUCTION