

# 

# Programmation système

# PROSIT 1 Aller :

# Burnout !

# Analyse du contexte

La société 2FRoaming permet de connecter les téléphones mobiles de leurs abonnés à leur opérateur pendant leur voyage à l’étranger.

Ils rencontrent des problèmes de méthode de travail, Mathilde est chargée de résoudre le problème en proposant un plan.

# Définition mots-clefs

Roaming : Le roaming fait référence à la capacité d'un utilisateur à se déplacer d'un réseau à un autre tout en restant connecté. Cela est souvent utilisé dans le contexte des réseaux mobiles où un utilisateur peut passer d'une cellule à une autre tout en maintenant la connectivité.

Steering of Roaming : Le "steering of roaming" se réfère aux stratégies mises en place pour diriger ou guider les utilisateurs d'un réseau à un autre de manière optimale, en fonction de divers critères tels que la qualité du signal, les coûts, etc.

Versioning : Le versioning est le processus de gestion des différentes versions d'un logiciel, d'un fichier ou d'un ensemble de données. Il permet de suivre les changements, de revenir à des versions antérieures et de gérer le développement de manière collaborative.

Gestion des builds : La gestion des builds concerne la création, la compilation et la validation d'une version logicielle à partir du code source. Cela inclut souvent l'automatisation du processus pour assurer la cohérence et la reproductibilité.

Tests unitaires : Les tests unitaires consistent à évaluer individuellement chaque composant ou fonction d'un logiciel pour s'assurer qu'ils fonctionnent correctement de manière isolée. Cela contribue à assurer la qualité du code et facilite la détection précoce des erreurs.

Git : Git est un système de gestion de versions décentralisé largement utilisé pour le suivi des modifications dans le code source pendant le développement logiciel collaboratif.

GitHub : GitHub est une plateforme de développement collaboratif basée sur Git. Elle offre des fonctionnalités telles que le stockage de code source, la gestion de versions, la collaboration et le suivi des problèmes.

Git Azure DevOps : Azure DevOps est une suite d'outils de développement proposée par Microsoft. Git est souvent utilisé dans le cadre d'Azure DevOps pour la gestion de versions et la collaboration dans le processus de développement.

IDE : IDE signifie "Environnement de Développement Intégré". Il s'agit d'un logiciel regroupant des outils essentiels pour le développement de logiciels, tels que l'éditeur de code, le débogueur et le compilateur.

Release Note : Une release note est une documentation qui accompagne la sortie d'une nouvelle version d'un logiciel. Elle contient des informations sur les nouvelles fonctionnalités, les corrections de bugs et d'autres détails pertinents.

Développement multi-plateforme : Le développement multi-plateforme consiste à créer des applications qui peuvent fonctionner sur plusieurs systèmes d'exploitation ou plates-formes, tels que Windows, macOS et Linux.

DevOps : DevOps est une culture et une approche de développement logiciel qui vise à améliorer la collaboration entre les équipes de développement et d'exploitation, en automatisant les processus et en favorisant une livraison continue.

Intégration continue : L'intégration continue est une pratique de développement où les modifications de code sont automatiquement testées et intégrées dans le projet principal dès qu'elles sont effectuées. Cela permet de détecter rapidement les problèmes d'intégration.

# Problématique

**- Quel plan d’amélioration proposer afin de répondre aux problématiques ?**

# Contraintes

- Développent Windows

- Développement multi-plateforme

- Compatibilité du plan indépendamment du nombre de collaborateurs

- Environnement Visual studio et Git

# Généralisation

Apprendre les bonnes pratiques de collaboration

# Livrables

Guide de bonnes pratiques pour le développement en équipe

Pistes de solution

Utiliser des branches pour chaque fonctionnalité

Conserver les versions modifiées / Proposer un guide des versions

Veiller à mettre en place des tests unitaires complet

Utiliser Sonar

# Plan d’action

Se renseigner sur les bonnes pratiques de collaboration

Documenter les différentes propositions proposées et leur alternative (comparatif)

Git …

# Réalisation du plan d’action

## Bonnes pratiques de collaboration :

### Communication transparente :

Assurez-vous que la communication au sein de l'équipe est claire et transparente. Utilisez des outils de messagerie instantanée, des réunions régulières et des plateformes de gestion de projet pour faciliter la communication.

### Définition des rôles :

Établissez clairement les responsabilités de chaque membre de l'équipe. Cela permettra d'éviter les malentendus et de maximiser l'efficacité.

### Utilisation d'outils de collaboration :

Choisissez des outils de collaboration adaptés aux besoins de l'équipe, tels que des plateformes de gestion de projet, des espaces de stockage partagés et des outils de suivi des versions.

### Réunions régulières :

Organisez des réunions régulières pour discuter des progrès, des défis et des ajustements nécessaires. Encouragez la participation de tous les membres de l'équipe.

### Gestion des versions :

Mettez en place une stratégie efficace de gestion des versions pour suivre les changements apportés au code source. Utilisez des outils de contrôle de version tels que Git.

## Documentation des propositions et alternatives :

### Documentation claire :

Pour chaque proposition, documentez de manière claire et concise les détails, y compris les objectifs, les avantages et les inconvénients.

### Comparatif des alternatives :

Fournissez un tableau comparatif des différentes propositions et de leurs alternatives. Incluez des critères tels que la performance, la maintenance, la scalabilité et la complexité.

### Feedback de l'équipe :

Encouragez les membres de l'équipe à donner leur feedback sur chaque proposition. Cela permettra d'obtenir des perspectives variées et d'améliorer la qualité des décisions.

### Historique des décisions :

Maintenez un historique des décisions prises, en mettant en évidence les raisons pour lesquelles une proposition a été choisie par rapport à d'autres alternatives.

### Réévaluation régulière :

Planifiez des moments réguliers pour réévaluer les choix techniques et les décisions prises, en tenant compte des changements de contexte et des nouvelles informations.

# Git

Git, un système de contrôle de version décentralisé, se révèle être un pilier essentiel dans le développement logiciel collaboratif. Utilisé pour suivre les modifications du code source, Git offre une structure flexible et efficace. GitHub, en tant que plateforme de gestion de projet basée sur Git, facilite la collaboration en fournissant des fonctionnalités telles que le suivi des problèmes, les demandes de tirage (pull requests), et la gestion des branches. Cette synergie entre Git et GitHub permet aux équipes de travailler de manière fluide, en synchronisant leurs contributions et en maintenant une cohérence dans le code. D'autre part, Git Azure DevOps, avec ses fonctionnalités intégrées pour le suivi des projets, la gestion de versions et l'intégration continue, offre une alternative complète. Son intégration transparente avec Azure D& evOps Services renforce la collaboration au sein d'équipes distribuées. Ainsi, que ce soit à travers GitHub ou Git Azure DevOps, l'utilisation judicieuse de Git dans le développement logiciel contribue à l'efficacité, à la traçabilité des changements, et à la gestion harmonieuse des projets.

# Sonar

SonarQube, dans le contexte du développement logiciel en équipe avec Git, GitHub, et Git Azure DevOps, joue un rôle crucial dans l'assurance de la qualité du code. SonarQube est une plateforme d'analyse statique du code qui identifie les défauts potentiels, les vulnérabilités de sécurité, et évalue la qualité globale du code source. Intégrer SonarQube dans le processus de développement permet d'automatiser l'inspection du code, offrant ainsi une rétroaction instantanée aux développeurs.

Que ce soit sur GitHub ou Git Azure DevOps, l'intégration de SonarQube peut être réalisée à travers des plugins dédiés. Lorsqu'une nouvelle modification est proposée via une demande de tirage (pull request), SonarQube analyse automatiquement le code concerné et génère des rapports détaillés sur les problèmes détectés. Cela permet à l'équipe de développement d'identifier et de résoudre rapidement les problèmes potentiels avant que le code ne soit fusionné dans la branche principale.

L'utilisation de SonarQube favorise la cohérence, la lisibilité, et la maintenabilité du code, contribuant ainsi à l'amélioration continue de la qualité logicielle. Cette synergie entre Git, GitHub, Git Azure DevOps, et SonarQube offre un environnement complet et robuste pour le développement collaboratif de logiciels de haute qualité.