Normes

NORMES

PROCÉDURES OU RÈGLES DE TRAVAIL

1. **Demander dès qu'on est bloqué**
2. **Éviter la répétition (toujours noter les solutions apportées dans le fichier de gestion des erreurs)**
3. **Exécution avant réclamation**
4. Faire le design thinking au démarrage d’un projet ou d’une fonctionnalité
5. Se référer à l”existant lors du démarrage de toute nouvelle fonctionnalité
6. Faire une réunion de lancement de semaine tous les lundis matin (durée : 10h à 11h)
7. Organiser la rétrospective à la fin de chaque sprint (toutes les deux semaines)
8. Ne démarrer aucune tache sans DoD (Definition of Done) sauf dérogation
9. Ne démarrer aucune tache sans délai
10. Pousser toujours la version de son travail chaque jour
11. Récupérer la dernière version stable tous les matins avant de démarrer le travail
12. Faire une présentation de chaque nouvelle fonctionnalité pour validation avant de demander la fusion ou la mise en production
13. Effectuer une démo des tâches réalisées dans la semaine tous les jeudis
14. Réaliser obligatoirement les tests postman et les tests unitaires pour l’écriture d’une API
15. Suivre les étapes suivantes pour la réalisation d’une tâche: **Implémenter l’API,Réaliser les interfaces validées, Consommer l’API**
16. Documenter tout nouveau concept implémenté dans un projet
17. Fixer un délai d'au plus 48 heures pour toutes les tâches
18. Avoir une fiche d’installation pour tous les projets
19. Renseigner les configurations nécessaires dans le fichier install.md du projet à chaque mise à jour du projet
20. Soumettre une demande de modification de la fiche d’installation pour une mise à jour
21. Utiliser la convention “Traduction_date” pour nommer les fichiers de traduction
22. Faire une mise en production tous les jeudis
23. Effectuer une mise en production les vendredis matin des nouvelles fonctionnalités validées et testées
24. Avoir toujours une branche [develop stable <version>] disponible sur le dépôt
25. Avoir toujours une branche [main stable <version>] disponible sur le dépôt

26. Avoir toujours une branche [prod stable <version>] pour la production disponible sur le dépôt.
27. Versionner le projet à chaque mise en production d'une nouvelle fonctionnalité ; Ex: v0.0.1
28. Faire une demande des assets au démarrage de tout projet mobile

NORMES APPLICATIVES

- Nommer les branches comme suit :
  - **"Setup/Nom Projet" (lors de l'initialisation du projet)**
  - **"Setup/Nom Config" (lors de la mise en place d'une configuration spécifique pour le projet),**
  - **"Feature/Api/Module" (lors de l'implémentation d'une API sur un module),**
  - **"Feature/Interface/Module" (lors de l'implémentation d'une vue sur un module),**
  - **"Feature/Consumption/Module" (lors de la consommation d'une API sur un module),**
  - **"Feedback/Api/Module" (lors de la correction d'une API existante sur un module),**
  - **"Feedback/Interface/Module" (lors de la correction ou mise à jour d'une interface sur un module),**
  - **"Feedback/Consumption/Module" (lors de la revue d'une consommation API sur un module)**
  - **"Feedback/Projet_date" (lors du traitement d'un fichier feedback)**
  - **"Module/Nom Module" (branche stable pour un module donné)**
  - **"Review/API/Module" (lors du code review)**

Nommer les commits de façon explicite par sous tâche (ajouter un titre et une description si nécessaire) .

ROUTINE QUOTIDIENNE DE TRAVAIL

- Coding Games (Heure de démarrage : 9H30 Durée: 15 min)
- Stand up meeting (Heure de démarrage :10h Durée: 15 min)
- Pause à 17h (Durée: 15 min)
- Point de la journée à 18h (Durée: 15 min)
- Rapport de semaine à envoyer tous les samedis au plus tard à 15h au manager de l'équipe :
  - ***Intitulé du rapport*** : RAP_KOB_*RB*_40 (RAP pour dire rapport , KOB les 3 premières initiales du nom du produit KOBO , RB pour Rachid Bouraima et 40 pour designer la semaine dans l'année) .
  - ***Point clé du rapport*** : Fait marquant ; point de blocage ; suggestion ou amélioration

- Retard de 15 min : signaler au manager de l'équipe et au lead tech
- Retard de plus de 15 min : signaler par mail au manager de l'équipe et mettre en copie l'administration

PRINCIPES ET BONNES PRATIQUES SUR LA QUALITÉ DE CODE

NB: Les fonctions et variables sont déclarées en anglais

1. Nom significatif et explicatif

- Déclarer les variables et les fonctions en anglais
- Donner des nom explicites aux variables et fonctions
- Utiliser le Pascal case pour les classes et les structures de données
- Utiliser le camelcase pour le nom de variable et fonctions
- Utiliser le snake case pour les constantes

2. Écriture des fonctions

- Typer les arguments et les retours
- Déclarer les variables en début de fonctions sauf en cas de porté de variable
- Écrire des fonctions courtes
- Indenter le code
- Eviter les magic number
- Limiter les arguments à 2 - 3 max
- Eviter les boolean flags
- Eviter Le conditionnel négatif
- Éviter le copier coller de fonctions

3. Principe SOLID

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle
- Dependency inversion principle

4. Règle boyscout (sous réserve d’avoir informé le manager de l'équipe)

"**Laisser le terrain de camping plus propre que vous ne l'avez trouvé en arrivant**"

NORMES DE CONCEPTION ET DE QUALITÉ D'UNE TACHE MOBILE REACT-NATIVE

1. Intégration d'une interface

- Avoir 90 % de conformité avec les interfaces figma et/ou les spécifications
- Utiliser le theming en place
- Eviter de mettre des couleurs et autres en dur dans le code
- Vérifier les arrondis, les alignements horizontales et verticales, les paddings et margins
- Mettre les bons textes
- Mettre en place les traductions
- Vérifier et optimiser l’accessibilité des différents composantes de l’interface
- Faire attention au nombre de re-render de l’interface
- Vérifier le bon affichage sur au moins trois marques de portable

2. Utilisation /Création d'un composant

- Respecter le règle de nommage en vigueur pour nommer et importer un composant
- Respecter le règle de nommage en vigueur pour nommer les paramètres(props) et les variables
- Respecter le nombre maximum de paramètres d’un composant (utilisé la destructuration au besoin)
- Faire attention au nombre de re-render du composant
- Externaliser tous les traitements de données hors du composant dans la mesure du possible
- Créer les composants dans le bon dossier et non sur l’interface elle même
- Pour les composants externes privilégiez ceux de la communauté et ceux proposé par la documentation officielle
- Utiliser les composants dédiés pour certain layouts (Ex : utilisé Flashlist au lieu de FlatList pour les listes scrollable)

3. Utilisation d'un package

- Utiliser les packages qu’en cas de réel nécessité
- Utiliser en priorité les packages de la communauté react-native
- Utiliser des packages non déprécier et dont la taille s’exprime en KB (pèse le moins)
- Utiliser les packages dédié et optimisé pour le mobile

4. Utilisation des animations

- Activer l’utilisation du driver pour toutes les animations

5.Général

- Ne pas laisser dans l’application des packages non utilisés
- Ne pas laisser dans l’application des variables non utilisés
- Ne pas laisser dans les composants et vues des imports non utilisés
- Supprimer tout composant et fichier non utilisés
- Enlever tous les codes commentés
- Enlever tous les console.log après développement
- Indenter le code avec un linter
- Régler au maximum les erreurs, warning de la console
- Ne pas faire une modification majeure sur un composant existant sans informer au préalable
- Utiliser les outils mise à disposition par react-native (hooks, context…)
- Utiliser au maximum les principes solides appliqués à react-natives

6. Livrable

- une branche nommée conformément aux règles de nommages
- un commit explicites de ce qui a été fait ou pas
- un fichier des variables de traduction à faire traduire s’il y en a

Définition Of Done d'une tache Front Web

- Intégrer les interfaces conformément aux maquettes (90%) ou spécifications fournies.
- Utiliser les conventions de nommage et de structuration du code définies par l'équipe.
- Mettre à jour les fichiers de configuration et de dépendances si nécessaire.
- Présenter un loader pour tous les chargements de données.
- Présenter un loader pour les boutons de validation de formulaire.
- Assurer un minimum de responsivité pour le format mobile.
- Inclure une pagination ou scroll pour les listes de données.
- Rendre les menus actifs.
- Implémenter la traduction des textes
- Indiquer les champs requis d’un formulaire par des marqueurs.
- Fournir un label et/ou un placeholder pour les champs d'un formulaire.
- Valider les zones et inputs de saisie.
- Formater les données suivant la langue. (montant , date ….)
- Utiliser adéquatement les composants.
- Garantir le fonctionnement optimal des composants.
- Gérer les réponses des appels API ou actions.

- Présenter une confirmation sur les actions critiques (suppression, etc.).
- Réaliser les tests unitaires pour les composants. (*)
- Effectuer les tests d'intégration pour vérifier l'interaction entre les composants.(*)
- Effectuer les tests manuels pour valider le bon fonctionnement de l'interface utilisateur et de la fonctionnalité
- Vérifier la compatibilité entre navigateurs (Chrome, Firefox, Safari, Edge, etc.).
- Vérifier et corriger les erreurs et les avertissements générés par les outils de linting et de formatage.
- Obtenir la validation des parties prenantes concernées (designers, product owners, etc.).
- Réaliser la revue de code par un autre développeur

Définition Of Done d'une tache Back

- Nommer correctement la branche de la tâche conformément aux normes applicatives définies.
- Écrire le code correspondant aux fonctionnalités prévues dans la User Story.
- Organiser au mieux le code (Model, Controller, Request, Resources, etc.).
- Respecter le Design Pattern MVC.
- Nommer correctement les variables et les fonctions conformément aux normes de développement définies.
- Appliquer le principe SOLID selon les normes de développement définies.
- Réaliser les traductions API.
- Respecter les hypothèses ou les conditions spécifiées dans la User Story.
- Effectuer les tests Postman.
- Réaliser les tests unitaires sans erreurs.
- Effectuer le formatage et le linting du code.
- Documenter les endpoints.
- Renseigner les configurations nécessaires dans le fichier install.md du projet à chaque mise à jour du projet.
- Effectuer la revue de code.