**Cette doc est à reprendre après remise à niveau de chatGPT**

**1. Simplification de l'installation avec un installateur Cockpit (.exe) et PowerShell**

Créer un **installateur .exe** pour le cockpit est une excellente idée pour faciliter l'installation, surtout pour des personnes qui n'ont pas l'habitude de manipuler des outils comme Git ou Node.js. L'idée d'intégrer PowerShell comme interface de communication est aussi pertinente, car elle permettrait de gérer certaines opérations en ligne de commande de manière plus transparente et contrôlée.

**Procédure d'installation simplifiée avec un installateur .exe :**

L'installateur .exe pourrait être conçu pour :

* **Télécharger et installer les dépendances automatiquement** : Git, Node.js, Visual Studio Code et autres outils nécessaires.
* **Cloner automatiquement le dépôt GitHub** : L'utilisateur n'aurait pas à se soucier de la configuration de Git ou de GitHub.
* **Installer et configurer le cockpit** : L'installateur pourrait automatiquement configurer l'application cockpit et la rendre prête à l'emploi.

**Utilisation de PowerShell :**

Intégrer PowerShell dans le cockpit peut permettre aux utilisateurs d'exécuter des commandes prédéfinies sans avoir à se plonger dans la ligne de commande complexe. Par exemple, PowerShell pourrait être utilisé pour :

* Lancer ou arrêter le serveur de développement.
* Exécuter des scripts de test.
* Vérifier la configuration de l'environnement.

PowerShell peut être inclus dans l'interface du cockpit, permettant ainsi une gestion simplifiée des processus.

**2. Détail de la gestion des branches et validation du code**

La gestion des branches est cruciale pour éviter les conflits lors du travail en équipe, et il est important de bien comprendre la procédure de validation pour que chaque membre puisse travailler sans perturber les autres. Voici une procédure détaillée :

**Procédure de gestion des branches pour validation :**

1. **Création d'une nouvelle branche** :
   * **Quand créer une branche ?** : Chaque collaborateur doit créer une nouvelle branche lorsqu'il commence à travailler sur une fonctionnalité ou une correction de bug.
   * **Comment créer une branche ?** :
     + Ouvrir GitHub Desktop.
     + Cliquer sur "Branch" > "New Branch".
     + Donner un nom à la branche (ex : feature/questionnaire-update ou bugfix/fix-button).
     + Cliquer sur "Create Branch" pour la créer et la basculer dessus.
2. **Faire les modifications sur la branche** :
   * Une fois sur la branche dédiée, le collaborateur peut effectuer les modifications nécessaires dans le code. Les modifications peuvent concerner la création d’un questionnaire, la correction d’un bug, ou l’ajout d’une fonctionnalité.
   * **Validation locale** : Avant de pousser les modifications sur GitHub, l’utilisateur doit s’assurer que tout fonctionne correctement localement (en testant l’application sur son PC, par exemple).
3. **Commit des modifications** :
   * Une fois les modifications effectuées, il faut les **commiter** avec un message clair.
   * Exemple de message de commit : "Ajout de la fonctionnalité de gestion des questions dans le questionnaire" ou "Correction du bug de validation des réponses".
4. **Pousser les modifications vers GitHub** :
   * Une fois le commit effectué, le collaborateur pousse les modifications vers GitHub via GitHub Desktop.
   * Cliquer sur "Push origin" pour envoyer les modifications sur la branche spécifique du dépôt distant.
5. **Revue de code par le responsable (moi ou un autre)** :
   * **Comment récupérer le code ?** : Le responsable ou un collègue doit récupérer la branche sur GitHub Desktop et la vérifier localement pour s'assurer que le code est fonctionnel et conforme aux attentes.
   * **Procédure de validation** :
     + **Télécharger la branche** en cliquant sur "Fetch origin" dans GitHub Desktop.
     + **Vérifier le code** : Tester la fonctionnalité ajoutée ou la correction du bug sur la branche locale pour s'assurer que tout fonctionne correctement.
     + **Commenter les bugs ou les problèmes** : Si des bugs ou des incohérences sont détectés, le responsable peut laisser des commentaires sur GitHub ou dans un fichier TODO pour indiquer les actions à entreprendre.
6. **Remontée des modifications après validation** :
   * Une fois que le code a été validé et que tous les commentaires ont été pris en compte, le collaborateur peut **remonter les modifications**.
   * Le collaborateur doit faire un **nouveau commit** en répondant aux commentaires ou en effectuant des corrections.
   * **Pousser à nouveau** les modifications validées vers GitHub.
7. **Fusion de la branche avec la branche principale (main)** :
   * Une fois la branche validée et prête à être intégrée, le collaborateur doit créer un **pull request** (PR) sur GitHub.
   * **Validation du pull request** : Le responsable ou un autre membre de l’équipe doit valider la PR, vérifier qu’il n’y a pas de conflits, et **fusionner** la branche avec la branche principale (main).
   * **Changer de branche** : Une fois la branche fusionnée, le collaborateur doit **revenir à la branche principale** et récupérer les dernières modifications en cliquant sur "Fetch" et "Merge".
8. **Nettoyage des branches** :
   * Une fois la branche fusionnée et que le travail est terminé, elle peut être **supprimée** localement et sur GitHub.
   * **Supprimer la branche** : Cela permet de garder le dépôt propre et d'éviter l’accumulation de branches inutiles.

**Résumé du processus de gestion des branches :**

* Créer une branche pour chaque tâche ou fonctionnalité.
* Faire des modifications et tester localement.
* Committer et pousser les modifications vers GitHub.
* Le responsable valide les modifications, fait des commentaires si nécessaire.
* Le collaborateur réintègre les corrections et pousse les changements.
* Fusionner la branche après validation et supprimer les branches terminées.