Guide de collaboration

Auteur : **Pavel KLIMOVICH**

Table des matières

1. Introduction

Ce document est essentiel à lire avant de commencer à travailler sur le projet ToDo&Co. Son contenu s'adresse principalement aux développeurs qui participeront tout au long de la durée du projet.

Deux aspects majeurs seront abordés : la collaboration sur Git ainsi que le maintien de la qualité du code conformément aux standards **PSR**.

La collaboration sur **Git** est un élément crucial pour assurer un développement fluide et efficace du projet. Git est un système de contrôle de version largement utilisé qui permet aux développeurs de travailler ensemble sur un même code source. Avant de commencer à contribuer au projet, assurez-vous de bien comprendre les concepts de base de Git, tels que les **branches**, les **commits**, les **pull requests**, et les conflits de fusion. Assurez-vous également de connaître les meilleures pratiques pour une utilisation efficace de Git, comme nommer correctement les branches, ajouter des messages de commit clairs et concis, et effectuer régulièrement des pulls et des merges pour rester synchronisé avec la branche principale.

En ce qui concerne la qualité du code, il est essentiel de respecter les standards **PSR-1**, **PSR-2**, **PSR-4** et **PSR-12**. Ces standards ont été établis par la PHP Standards Recommendation (PSR) pour garantir une cohérence et une lisibilité accrues du code PHP. Assurez-vous de suivre ces standards lors de l'écriture de votre code, en respectant les conventions de nommage, l'indentation, l'utilisation des espaces et des accolades, ainsi que les bonnes pratiques en matière de commentaires. L'utilisation d'un outil d'analyse statique du code, tel que PHP_CodeSniffer, peut vous aider à détecter les violations des standards et à les corriger rapidement. Vous pouvez également consulter la documentation de Symfony sur les standards mises en place par le framework.

En résumé, avant de vous plonger dans le projet ToDo&Co, assurez-vous de maîtriser Git et de comprendre comment collaborer efficacement avec l'équipe de développement. De plus, respectez les standards PSR-1, PSR-2, PSR-4 et PSR-12 pour maintenir la qualité du code à un niveau élevé. En suivant ces bonnes pratiques, vous contribuerez de manière significative au succès du projet.

1. Utilisation du GIT

Dans le but de maintenir une traçabilité des différentes évolutions apportées tout au long de la durée du projet et de disposer de mécanismes efficaces de retour en arrière, nous avons choisi d'utiliser GitHub comme solution d'hébergement et de partage du code.

GitHub est une plateforme de développement collaboratif qui offre de nombreuses fonctionnalités pour faciliter la gestion de versions et la collaboration entre les développeurs. En utilisant GitHub, nous pourrons créer un référentiel (repository) dédié au projet ToDo&Co, où nous pourrons stocker et suivre toutes les modifications apportées au code source.

L'utilisation de GitHub présente plusieurs avantages. Tout d'abord, il permettra à chaque membre de l'équipe de travailler sur une branche séparée, ce qui favorise le développement parallèle de fonctionnalités sans perturber le code principal. Ensuite, GitHub propose un système de contrôle de version avancé, qui enregistre chaque commit effectué, facilitant ainsi la récupération de versions antérieures en cas de besoin de rollback. De plus, les fonctionnalités de **pull requests** de GitHub permettent une revue de code efficace et une intégration fluide des contributions de chaque développeur dans le code principal.

En utilisant GitHub comme solution d'hébergement et de partage du code, nous veillerons à ce que toutes les évolutions apportées au projet ToDo&Co soient tracées et documentées de manière transparente. Cela facilitera la collaboration entre les membres de l'équipe et permettra une gestion efficace des versions et des retours en arrière si nécessaire.

1. Récupération du projet

Assurez-vous d'avoir Git installé sur votre machine. Si ce n'est pas déjà fait, vous pouvez le télécharger et l'installer à partir du site officiel de Git.

Ouvrez votre terminal ou votre interface de ligne de commande et naviguez vers le répertoire dans lequel vous souhaitez cloner le dépôt du projet ToDo&Co.

Une fois dans le répertoire approprié, utilisez la commande git clone suivie de l'URL du dépôt pour effectuer le clonage. L'URL du dépôt vous sera fournie par l'équipe de développement. Voici un exemple de commande :

git clone <URL-du-dépôt>

Une fois la commande terminée, vous aurez récupéré avec succès le code du projet ToDo&Co sur votre machine. Vous pouvez maintenant commencer à travailler sur le projet en utilisant les fichiers clonés localement.

2. Modifications

Pour pouvoir effectuer vos modifications sans impacter le travail des autres développeurs intervenant actuellement sur le projet, la gestion des modifications s'effectue de la manière suivante :

a. Création d'une branche de feature

Pour créer une nouvelle branche de fonctionnalité sur laquelle vous pourrez effectuer vos modifications sans modifier la branche principale. Pour créer une nouvelle branche de fonctionnalité, utilisez la commande **git checkout -b** suivie du nom de votre branche. Par exemple, si vous souhaitez créer une branche nommée "**my-feature-name**", exécutez la commande suivante :

git checkout -b my-feature-name

Vous pouvez maintenant commencer à effectuer vos modifications dans cette nouvelle branche. Toutes les modifications que vous effectuerez et tous les commits que vous ferez seront enregistrés dans cette branche spécifique, sans affecter la branche principale.

b. Enregistrer les modifications sous forme de commits

Lorsque vous avez terminé une tâche répertoriée dans les issues du projet, vous pouvez enregistrer vos modifications sous forme de commit sur votre branche en indiquant l'issue associée à ce commit.

Pour voir la liste des modifications que vous avez apportées, vous pouvez utiliser la commande ***git status***. Cela affichera les fichiers modifiés et les fichiers non suivis.

Ajoutez les fichiers modifiés à l'index en utilisant la commande ***git add***. Par exemple, si vous souhaitez ajouter tous les fichiers modifiés, exécutez la commande suivante :

git add .

Une fois que vous avez ajouté les modifications à l'index, vous pouvez créer un commit en utilisant la commande **git commit**. Assurez-vous d'inclure un message de commit clair et descriptif qui indique l'issue clôturée par ce commit. Par exemple :

git commit -m "Fix issue #123: Implement feature XYZ"

Remplacez "#123" par le numéro réel de l'issue que vous avez clôturée et "Implement feature XYZ" par une description concise de la fonctionnalité ou du correctif que vous avez apporté.

Cela enverra vos commits et modifications locales vers le dépôt distant associé à votre branche de fonctionnalité spécifique.

c. Enregistrer vos modifications dans le dépôt distant

Pour enregistrer vos modifications dans le dépôt distant et permettre aux autres développeurs d'y accéder si nécessaire, vous devez effectuer un "**push**" de vos modifications.

Avant de pousser vos modifications, assurez-vous que vous êtes à jour avec les derniers changements du dépôt distant en effectuant un "**git pull**" (récupération) des dernières modifications.

Une fois que vous êtes à jour, vous pouvez pousser vos modifications vers le dépôt distant en utilisant la commande **git push**. Par exemple, si votre branche de fonctionnalité s'appelle "*my-feature-name*", exécutez la commande suivante :

git push origin my-feature-name

Il est recommandé de toujours effectuer un "pull" avant de pousser vos modifications pour vous assurer que vous travaillez sur la dernière version du code du dépôt distant et éviter les conflits de fusion.

En poussant régulièrement vos modifications vers le dépôt distant, vous maintenez une sauvegarde en ligne de votre code et facilitez la collaboration avec d'autres développeurs sur le projet.

d. Effectuer un pull request

Une fois que vous avez terminé de travailler sur l'ensemble des issues qui vous ont été affectées, vous pouvez soumettre votre code à la validation des autres développeurs du projet en créant un **pull request**. Voici les étapes à suivre :

1) Assurez-vous que toutes vos modifications sont enregistrées sous forme de commits et poussées sur le dépôt distant, comme décrit précédemment.
2) Accédez à la page du dépôt du projet sur GitHub.
3) Sur la page du dépôt, recherchez le bouton "Pull requests", il se trouve près de la barre de navigation supérieure.
4) Sur la page des pull requests, cliquez sur le bouton "New pull request" pour créer un nouveau pull request.

5) Vous serez dirigé vers la page de création d'un nouveau pull request. Sélectionnez la branche que vous souhaitez fusionner dans le menu déroulant "base branch" (branche de base). Cela représente généralement la branche principale du projet, comme "main".
6) Dans le menu déroulant "compare branch" (branche à comparer), sélectionnez votre branche de fonctionnalité qui contient vos modifications.
7) Une fois les branches sélectionnées, ajoutez un titre clair et descriptif à votre pull request. Vous pouvez également ajouter un commentaire supplémentaire expliquant les modifications que vous avez apportées.
8) Si nécessaire, vous pouvez ajouter des assignés ou des relecteurs (reviewers) à votre pull request. Ces personnes seront notifiées de votre demande et pourront examiner et valider votre code.
9) Lorsque vous avez terminé de remplir les informations de la pull request, cliquez sur le bouton "Create pull request" pour soumettre votre demande.

Votre **pull request** sera désormais visible pour les autres développeurs du projet. Ils pourront passer en revue votre code, laisser des commentaires et, éventuellement, approuver la fusion de votre branche de fonctionnalité avec la branche principale.

Il est important de noter que la création d'un pull request permet d'ouvrir une discussion et de faciliter la collaboration entre les développeurs pour valider et intégrer les modifications de manière transparente dans le projet.

e. - Intégration de vos modifications à la branche main

Une fois que votre **pull request** a été revue et validée par vos pairs, elle peut être fusionnée directement dans la branche principale du projet. Cette fusion intégrera vos modifications fonctionnelles et de qualité de code à la version principale du projet.

3. Qualité de code

Pour valider le respect des standards mise en place, il est demandé à chaque développeur Intervenant sur le projet de procéder à une vérification automatisée de son code grâce à ces outils :

1/ CodeSniffer

C'est outil de vérification de code source pour PHP. Il permet d'appliquer des normes de codage prédéfinies et de détecter les violations de ces normes dans le code PHP. Il peut également détecter les erreurs de formatage, les conventions de nommage non respectées, les indentations incorrectes, les espaces en trop ou manquants, et bien d'autres problèmes de style et de structure de code.

Vous pouvez vérifier votre code en exécutant la commande suivante :

./vendor/bin/phpcs --report-full=QualityReport.txt --ignore=/var,/vendor

Il ne vous reste donc plus qu'à corriger les erreurs remontées par le rapport en question avant de push votre branche.

Pour plus de l'information sur l'utilisation de CodeSniffer vous pouvez consulter la documentation officielle.

2/ PHPStan

C'est un outil d'analyse statique pour PHP. Il permet de détecter les erreurs et les problèmes potentiels dans votre code PHP sans avoir besoin de l'exécuter. Contrairement à PHP_CodeSniffer qui se concentre sur les règles de codage et le style, PHPStan se concentre principalement sur la détection des erreurs de logique, des types incorrects, des problèmes de compatibilité et d'autres erreurs potentielles qui pourraient se produire lors de l'exécution du code.

Vous pouvez vérifier votre code en exécutant la commande suivante :

vendor/bin/phpstan analyse -l 5 src tests

Pour plus de l'information sur l'utilisation de PHPStan vous pouvez consulter la documentation officielle.