# Contribuer à `ToDo List`

En tant que contributeur, voici les lignes directrices que nous aimerions que vous suiviez :



## Question ou Problème

N'ouvrez pas d'Issue pour des questions d'assistance générale car nous voulons conserver les Issues GitHub pour les rapports de bogues et les demandes de fonctionnalités.

Contactez l'assistance de ToDo & Co.

## Issues et Bugs

Si vous trouvez un bug dans le code source, vous pouvez nous aider en soumettant une Issue à notre dépôt GitHub. Mieux encore, vous pouvez soumettre une Pull Request avec un correctif.

## Demande de fonctionnalité

Vous pouvez demander une nouvelle fonctionnalité en soumettant une Issue à notre dépôt GitHub. Si vous souhaitez implémenter une nouvelle fonctionnalité, veuillez d'abord soumettre une Issue avec une proposition pour votre travail, pour être sûr que nous pouvons l'utiliser. S'il vous plaît, réfléchissez à quel genre de changement il s'agit :

- Dans le cas d'une fonctionnalitée majeur, ouvrez d'abord une Issue et décrivez votre proposition afin qu'elle puisse être discutée. Cela nous permettra également de mieux coordonner nos efforts, d'éviter le dédoublement du travail et de vous aider à concevoir les changements pour qu'ils soient acceptés avec succès dans le projet.
- Les petites fonctions peuvent être soumises directement par une Pull Request. En cas de doute soumettez une Issue.

## Guide de soumission

### Workflow

Le projet ToDo List utilise le workflow Gitflow popularisée par Vincent Driessen (nvie).

- Pour la correction d'un bug créez une branche `hotfix\My-Issue` depuis la branche `master`.
- Pour la soumission d'une nouvelle fonctionnalitée, créez une branche `feature\My-Issue` depuis la branche `develop`.

### Soumettre une Issue

Avant de soumettre une Issue, veuillez effectuer une recherche dans l'outil de suivi des Issues, peut-être qu'il existe déjà une Issue pour votre problème et que la discussion pourrait vous informer des solutions disponibles.

Nous voulons corriger tous les problèmes dès que possible, mais avant de corriger un bogue, nous devons le reproduire et le confirmer. Afin de reproduire les bogues, nous vous demanderons systématiquement de fournir les informations suivantes :

- Version de ToDo List utilisée
- Bibliothèques tierces et leurs versions
- Et le plus important - un cas d'utilisation qui échoue

Malheureusement, nous ne sommes pas en mesure d'enquêter / corriger les bugs sans un cas d'utilisation qui échoue, donc si nous n'avons pas de nouvelles de vous, nous fermerons une Issue qui n'a pas assez d'informations pour être reproduite.

## Soumettre une Pull Request

Avant de soumettre votre Pull Request (PR), tenez compte des lignes directrices suivantes :

1. Recherchez sur GitHub pour une PR ouverte ou fermée qui se rapporte à votre soumission. Vous ne voulez pas dupliquer l'effort.
2. Assurez-vous qu'une Issue décrit le problème que vous corrigez ou documente la conception de la fonctionnalité que vous souhaitez ajouter. En discutant de la conception à l'avance, nous nous assurons d'être prêts à accepter votre travail.
3. Fork du repositorie ToDo List.
4. Faites vos changements dans une nouvelle branche git :

   ```
   git checkout -b hotfix/My-Issue master
   ```

5. Créez votre patch, y compris les tests.
6. Suivez nos règles de codage.
7. Exécutez la suite de tests complète, et assurez-vous que tous les tests réussissent.
8. "Commit" vos modifications à l'aide d'un message de validation descriptif qui respecte les conventions de notre guide des messages de Commit.

   ```
   git commit -a
   ```

9. Poussez votre branche vers GitHub :

   ```
   git push origin fix/My-Issue
   ```

10. Dans GitHub, envoyez une Pull Request vers `ToDoList:develop`.

Si nous suggérons des changements, alors :

- Effectuez les mises à jour nécessaires.
- Ré-exécutez les suites de tests pour vous assurer que les tests sont toujours valide.
- Rebasez votre branche et forcez le push vers votre dépôt GitHub (cela mettra à jour votre Pull Request) :

```
git rebase master -i
git push -f
```

C'est bon! Merci pour votre contribution!

**Après la fusion de votre Pull Request**

Vous pouvez supprimer votre branche en toute sécurité :

- Supprimez la branche distante sur GitHub :

```
git push origin --delete feature/My-Issue
```

- Basculer la branche maître :

```
git checkout master -f
```

- Supprimer la branche locale :

```
git branch -D feature/My-Issue
```

- Mettez à jour votre branch master/develop avec la dernière version :

```
git pull --ff upstream master
```

# Règle de Codage

Pour assurer la cohérence du code source, gardez ces règles à l'esprit lorsque vous travaillez :

- Toutes les fonctionnalités ou corrections de bogues doivent être testées par un ou plusieurs tests (tests unitaires et/ou fonctionnels).
- Nous respectons le guide de style PSR1, PSR2 et Symfony. Veuillez utiliser l'outil php-cs-fixer pour vérifier votre code.

Exemple de commande:

```
php-cs-fixer fix src --rules=@Symfony,@PSR1,@PSR2
```

## Guide des messages de Commit

Les messages de Commit sont structurés de la manière suivante:

```
<type>(<scope>) : <description>
```

Le scope est facultatif.

1. **Type** :
    - **build** : Changements qui affectent les dépendances externes (par exemple : composer require/remove etc...)
    - **ci** : Modifications apportées aux fichiers de configuration (exemples de portée : Travis, Codacy, etc...)
    - **doc** : Modifications de la documentation uniquement
    - **feat** : Une nouvelle fonctionnalitée
    - **fix** : Correction d'un bug
    - **perf** : Un changement de code qui améliore les performances
    - **refactor** : Un changement de code qui ne corrige pas un bogue et n'ajoute pas de fonctionnalité
    - **style** : Changements qui n'affectent pas la signification du code (espace blanc, formatage, points-virgules manquants, etc.)
    - **test** : Ajout de tests manquants ou correction de tests existants
2. **Scope** : (facultatif)
    - **form** : Changement affectant les formulaires
    - **controller** : Changement affectant les controllers
    - **config** : Changement affectant les fichiers de configuration
    - **entity** : Changement affectant les entités
    - **service** : Changement affectant les services
    - **security** : Changement affectant la sécurité (security.yml, annotation, ...)
    - **fixture** : Changement affectant les fixtures
    - **repo** : Changement affectant les repository
    - **listener** : Changement affectant les listeners
    - **command** : Changement affectant les lignes de commandes php(cli)
    - **view** : Changement affectant les vues de l'application

3. **Description** : Le sujet contient une description succincte du changement

    - utiliser l'impératif du présent : "change" pas "changed" ni "changes"
    - ne mettez pas la première lettre en majuscule
    - pas de point (.) à la fin