Contributing Guide sur Github

Installation du projet

Pour contribuer au projet:

- forker le projet sur votre machine pour cloner le repository sur votre compte Github.
- Suivez le fichier README.md pour installer le projet.
- Créez une nouvelle branche pour la fonctionnalité et positionnez vous sur cette branche.
- Codez la nouvelle fonctionnalité ou bugfix.

Modification

Toutes modification devra être testée avec PhpUnit. Les tests devront être executés avec la commande `php bin/phpunit`

Voici les commandes liées à phpunit :

- `vendor\bin\simple-phpunit` Lance tous les tests
- `vendot\bin\simple-phpunit <NomDuFichier>.php` Lance tous les tests d'un fichier
- `vendor\bin\simple-phpunit --filter <NomDeLaMéthode>` Test d'une méthode spécifique
- `vendor\bin\simple-phpunit --coverage-html web\test-coverage` Coverage généré par xDebug

Quand vos modification sont terminés et que les tests sont valides, vous pouvez soumettre votre pull request et attendre qu'elle soit acceptée.

Les règles à respecter

Respect des normes PSR-1 / PSR-12 / PSR-4. Vérifiez les bonnes pratique de Symfony

Contributing Guide sur GitLab avec intégration continue

(le repository GitHub est automatiquement cloné lors des push sur GitLab)

- Demandez un accès de membre sur le repository Gitlab.
- Ajoutez votre clé SSH dans vos settings de compte sur GitLab. (Votre passphrase sera nécessaire pour chaque push que vous ferez).
- Cloner le projet sur votre machine.
- Suivre les mêmes instructions que pour la contribution sur Github (branche, code).

Tests et Analyse du code

Les parties tests Unitaires/Fonctionnels ainsi que l'analyse du code sont entièrement pris en charge sur Gitlab grâce à GitLab CI (Continuous Integration)

Un fichier de script `.gitlab-ci.yml` a été ajouté à la racine du repository.

Lors d'un push sur le repository ce fichier sera détecté et lancera une pipeline que vous pourrez suivre en temps réel.

Ce fichier determine les étapes que va suivre la pipeline et les images dont il a besoin :

Premiere étape : CodingStandards

- `SecurityChecker` Outil de sensiolabs qui va vérifier si votre application utilise des dépendances avec des vulnérabilités de sécurité connues.
- `PHP_CodeSniffer` détecte les violations de code sur une norme spécifique (PSR-12 ici).
- `phpstan` détecte les erreurs dans le code.
- `twig-lint` vérifie la syntaxe des fichiers twigs.

Deuxième étape : BuildAssets

- Cette étape build les assets de l'application (évitant les erreurs lors des tests fonctionnels) et les archives pour le job suivant.

Troisième étape : PHPUnit

- Le job installe les dépendances nécessaires (mysql, composer, pdo, etc...).
- Mise en place de la BDD et chargement des fixtures pour les tests:

```
- php bin/console doctrine:database:drop --force --env=test
- php bin/console doctrine:database:create --env=test
- php bin/console doctrine:migration:migrate --env=test --no-interaction
- php bin/console doctrine:fixtures:load -n --env=test
```

- lancement de la commande phpunit pour executer les tests:
  `php bin/phpunit`

Vous pourrez suivre l'état d'avancement de la pipeline et vérifier si les jobs sont validés.

Une fois tous les éléments validés vous pouvez soumettre une merge request.