

# Audit de qualité du code et performance de l'application

## Qualité du code :

### 1. Mise à jour du framework :

Le projet était à l'origine sous la version de Symfony 3.1 .

Une mise à jour du framework à été faite pour garantir une version stable et à jour de l'application.

Désormais le projet est sous la version 5.0.7 de Symfony.

```
Symfony 5.0.7 (env: dev, debug: true)
```

### 2. Minification CSS et JS :

Plusieurs fichiers CSS et JS sont présents sur le projet.

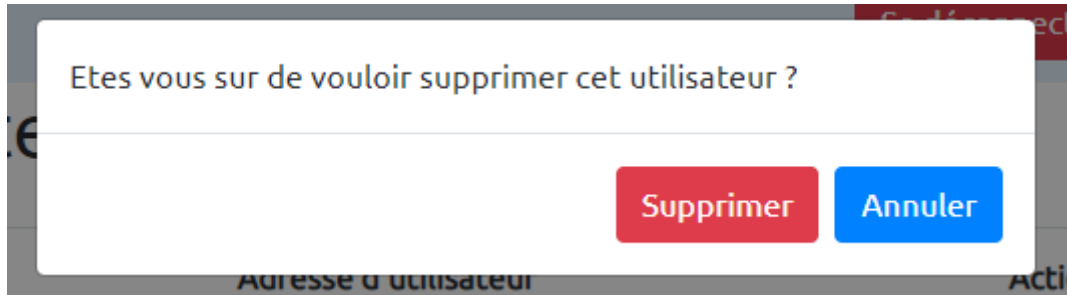
Ces fichiers ont été minifiés afin de garantir une rapidité d'exécution optimale lors des chargements.

```
</div>
<!------->
<script src="{{ asset('../public/vendor/jquery/jquery-3.2.1.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/js/bootstrap.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/fonts/fontawesome-free-5.13.0-web/js/all.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/vendor/ansition/js/ansition.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/vendor/select2/select2.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/vendor/daterangepicker/moment.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/vendor/daterangepicker/daterangepicker.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/vendor/countdowntime/countdowntime.min.js') }}"></script>
<!------->
<script src="{{ asset('../public/js/main.js') }}"></script>
<!------->
<!------->
<link rel="stylesheet" type="text/css" href="{{ asset('../public/css/main.min.css') }}">
<!------->
```

### 3. Autres modifications :

Des modals de validation ont été ajoutés lors de l'action de suppression d'un utilisateur ou d'une tâche.

Cela permet d'éviter les actions involontaire lors de l'utilisation de l'application.

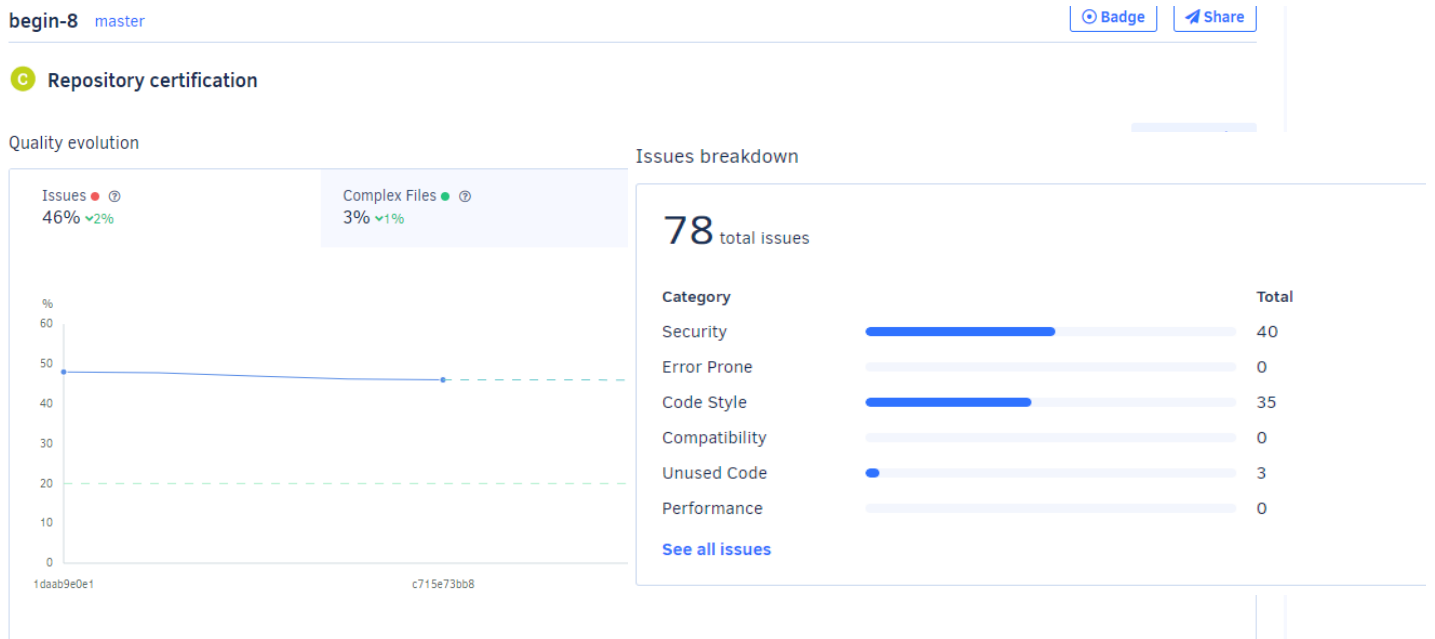


### 4. Codacy :

Grâce à l'outil Codacy, la qualité du code à pu être évalué sur différents critères.

Il permet de lister toutes les erreurs et problèmes dans le code du projet.

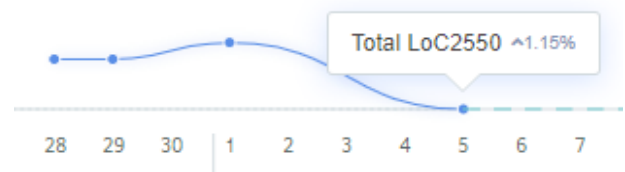
Lors de la première analyse à la reprise du projet, codacy à donné une note C au projet, avec un total de 78 issues, dont 40 relatif a des problèmes de sécurité.



Après l'analyse du 1er Mai, et l'ajout de différentes fonctions dans le projet, le nombre d'issues passe à 19, et 10 erreurs de code inutilisé.



En date du 5 Mai, lors de la dernière analyse et après corrections des erreurs précédentes, le code ne contient plus aucune erreur.



Codacy attribue une note au projet pour définir la qualité du code.  
Dès lors que le projet ne contient plus d'erreur, il obtient la note de A.

OC\_Projet\_8 master

**A** Repository certification

Ci-dessous quelques exemples de corrections relatives aux variables inutilisées détectées par Codacy.

```

1  src/Controller/DefaultController.php
@@ -10,7 +10,6 @@ class DefaultController extends AbstractController
10  10
11  11     public function indexAction()
12  12     {
13  13         $user = $this->getUser();
14  13         return $this->render('default/index.html.twig');
15  14     }
16  15 }
  
```

```

2  src/Controller/SecurityController.php
@@ -10,7 +10,7 @@
10  10     class SecurityController extends AbstractController
11  11     {
12  12
13  13         public function loginAction(AuthenticationUtils $authenticationUtils, Request $request)
14  13         public function loginAction(AuthenticationUtils $authenticationUtils)
15  15     {
16  16         $error = $authenticationUtils->getLastAuthenticationError();
  
```

## 5. Tests fonctionnels et tests unitaires :

Des tests fonctionnels et unitaires ont été implémentés afin de vérifier le bon fonctionnement de l'application lors de chaque modifications.

Vous pouvez exécuter ces tests grâce à la commande « php bin/phpunit --coverage-html tests/Rapports ».

Si tous les tests se sont bien déroulés, vous obtiendrez alors une réponse verte.
















```
Testing Project Test Suite
.....

Time: 27.63 seconds, Memory: 70.00 MB

OK (32 tests, 82 assertions)
```

A la suite de ces tests, vous pouvez également obtenir des résultats plus détaillés en vous rendant dans le dossier tests/Rapport et en ouvrant le fichier index.html.

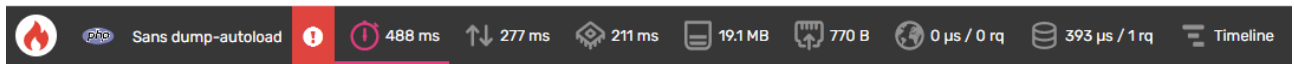
Sur cette page, le pourcentage de couverture des tests, appelé coverage, est affiché.

	Code Coverage							
	Lines		Functions and Methods			Classes and Traits		
Total		83.64% 184 / 220		83.67% 41 / 49		75.00% 9 / 12		
■ Controller		83.33% 110 / 132		53.85% 7 / 13		50.00% 2 / 4		
■ Entity		100.00% 39 / 39		100.00% 25 / 25		100.00% 2 / 2		
■ Form		100.00% 15 / 15		100.00% 3 / 3		100.00% 3 / 3		
■ Repository		100.00% 14 / 14		100.00% 4 / 4		100.00% 2 / 2		

## Performances :

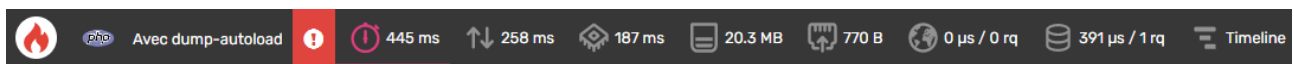
L'outil Blackfire a été utilisé pour analyser l'application, et mesurer les performances de celle-ci. Blackfire nous donne également des axes d'amélioration, afin de garantir des performances optimales pour l'utilisateur.

Lors de la première analyse, sans modification, les résultats obtenus étaient les suivants :



Blackfire a alors recommandé de créer un autoload grâce à la commande « `composer dump-autoload -o` », qui permet d'accélérer le chargement des pages.

Suite a cette modification, voici les résultats obtenus ainsi que le comparatif des deux analyses :

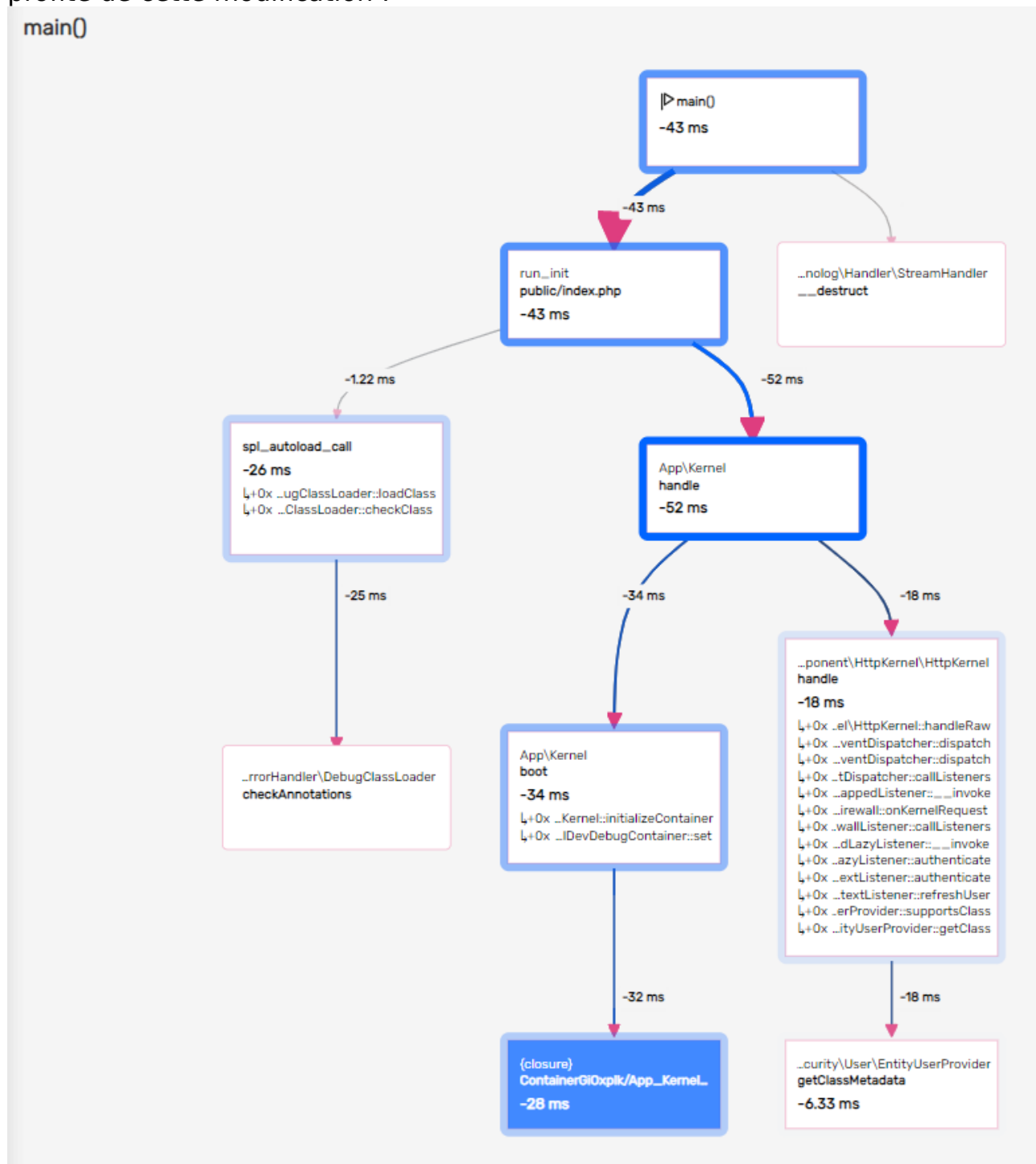


Comparaison :



On peut voir une nette amélioration du temps de chargement, de L'I/O wait et du chargement CPU, mais également une augmentation de la mémoire utilisée, qui montre bien que l'autoload est maintenant actif.

Le graphique de comparaison montre également quelle branche à le plus profité de cette modification :



Le bleu sur ce graphique montre les fichiers qui ont profité de cette amélioration, ainsi que le temps gagné entre les deux analyses.

## Bilan :

Grâce aux différentes modifications apportées à l'application, le projet a pu être considérablement amélioré sur l'aspect qualité et performance.

La note de qualité de Codacy a évolué de la lettre « C » à la lettre « A ».

Les performances analysées par BlackFire ont augmenté de 7 à 11%.

Le code a été simplifié et standardisé, ce qui permettra à d'autres développeurs de travailler sur le projet sans problème.

L'expérience utilisateur a également été simplifiée et améliorée.