

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

Ce document va vous présenter les différents éléments constituant la dette technique de l'application qui devront être corrigé par la suite pour garantir un haut niveau de performance.

## Qualité du code

### Version obsolète de Symfony

La version actuelle du projet est Symfony 3.2, cette version ne fait actuellement plus l'objet d'un support par la société gérée par Symfony, les bugs éventuellement présent ne feraient donc pas l'objet de correction ce qui pourrait notamment poser des problèmes de sécurité.

Il serait donc préférable de faire évoluer le projet vers une version de Symfony faisant l'objet d'un support, soit les versions 3.4 ou la version 4.0.

### Fichier absent du projet

Le fichier JQuery est référencé dans la vue **base.html.twig** mais n'est pas inclus dans le projet. L'absence du fichier JQuery prive le projet de la possibilité d'utiliser une partie des fonctionnalités offerte par Bootstrap qui est utilisé dans ce projet.

Il faudra donc inclure un fichier JQuery dans le projet.

### Dépréciation des méthodes lors de la validation des formulaires

Depuis la version 3.2 de Symfony, il est déprécié de contrôler la validité d'un formulaire sans avoir contrôlé que le formulaire a bien été envoyé. Faire ceci dans la version 4 de Symfony déclencherait une erreur.

Pour y remédier il faudra changer dans les actions relatives à la validation des formulaires ce code :

```
if ($form->isValid())
```

Et les remplacer par ce code

```
if ($form->isSubmitted() && $form->isValid())
```

### Variable non utilisée

L'analyse via Codacy révèle la présence de variable inutilisée dans le code de cette application. Par souci de clarté du code il est bon d'éviter la présence de variable inutilisée et donc inutile.

Exemple de l'action loginAction du SecurityController :

```

public function loginAction(Request $request)
{
    $authenticationUtils = $this->get('security.authentication_utils');

    $error = $authenticationUtils->getLastAuthenticationError();
    $lastUsername = $authenticationUtils->getLastUsername();

    return $this->render('security/login.html.twig', array(
        'last_username' => $lastUsername,
        'error'         => $error,
    ));
}

```

Dans ce cas, la variable **\$request** n'est pas utilisé dans le code de l'action et devrait donc être retiré.

### Sécurité du mot de passe insuffisante

L'entité User qui gère les utilisateurs ne comporte pas d'attribut salt. Hors le salt permet de renforcer le cryptage du mot de passe de manière significative en donnant une clé de cryptage unique à chaque mot de passe. En l'absence de salt, on peut considérer le cryptage comme étant insuffisant.

Pour résoudre ce problème vous devez ajouter un salt à l'entité User et veillez à ce que chaque utilisateur bénéficie d'un salt unique généré aléatoirement.

### Absence de message d'erreur adéquat

Lors de la création d'un utilisateur les contraintes d'unicités sur les emails et sur les noms d'utilisateur sont bien présentes mais ces contraintes ne sont pas associées à des messages clairs rendant compte de l'erreur à l'utilisateur lors de l'envoi du formulaire. En l'état, en cas de l'erreur un message standard est affiché.

Pour résoudre ce problème, il faudra ajouter des messages personnalisés pour ces erreurs dans la classe User.

### Bouton non utilisé

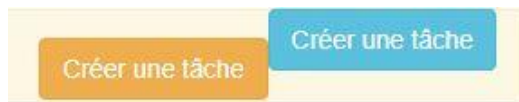
Le bouton « Consulter la liste des tâches terminées » n'est pas utilisé et ne renvoie vers aucun lien. De plus il n'y a aucune action dans le code de l'application qui corresponde au libellé de ce bouton. Il est également à noter que les tâches terminées ou en cours ne sont pas séparées et toutes accessibles depuis le bouton « Consulter la liste des tâches à faire » ce qui est incohérent avec la présence de deux boutons.

Pour résoudre ce problème, il y a deux possibilités :

- Avoir un seul bouton avec un libellé cohérent indiquant la présence de toutes les tâches
- Avoir deux boutons renvoyant chacun vers des listes de tâches distinctes

### Mauvais positionnement des boutons dans la liste des tâches

Quand la liste des tâches ne contient aucune tâche il y a 2 boutons de créations d'une tâche côte à côte qui sont mal alignés.



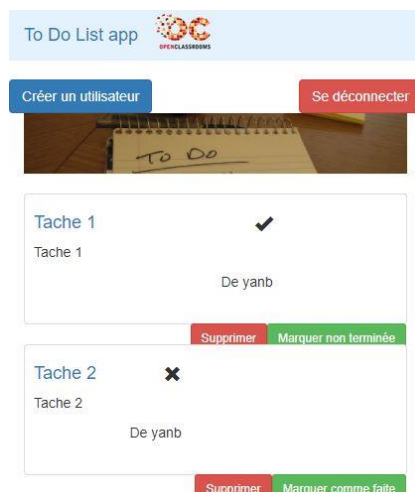
Quand la liste contient au moins 3 tâches, le bouton de création de tâche n'est plus correctement aligné.



Pour résoudre ce problème, il faudra supprimer l'un des boutons de création d'une tâche en cas de liste vide. Dans le cas d'une liste contenant au moins 3 tâches, il faudra veiller à ce que le bouton de création d'une tâche ne vienne pas casser l'alignement.

### Affichage défilant de la liste des tâches sur mobile

L'affichage de la liste des tâches sur mobile souffre de quelques problèmes. D'abord les boutons sont en partie recouverts par la tâche présente en dessous. Ensuite l'affichage des tâches est asymétrique selon que la tâche soit terminée ou non et dans aucun de ces deux cas l'affichage n'est correct car il laisse une large partie de l'élément dédié à la tâche vide.



Pour résoudre ces problèmes il faudra revoir le style lié à ces éléments via les classes CSS. Il faudra veiller à ce que les boutons ne soit pas recouvert et à ce que l'affichage des tâches soit le même et que les différents éléments de la tâche occupe l'ensemble de l'espace qui leur est dédié.

### Navigation difficile

Le site web ne contient pas de menu accessible sur toutes les pages du site, ce qui rend parfois extrêmement difficile la navigation, il par exemple pas possible depuis la page d'édition d'une tâche de revenir à la listes des tâches avec les liens fourni par le site web.

Pour résoudre ce problème, il faudrait d'abord ajouter un menu avec des liens permettant de se rendre très facilement sur les principales pages du site web, ensuite il faudra ajouter des liens sur certaine page permettant de revenir à une page parente (Exemple : depuis les formulaires de création et d'édition d'une tâche il doit être possible de revenir sur la page listant les tâches).

### Mauvaise organisation du formulaire d'édition d'un utilisateur

Le formulaire d'édition d'un utilisateur ne sépare pas la modification du mot de passe de la modification du reste des informations. Cela crée une difficulté, car pour modifier une information il faut entrer son mot de passe à deux reprises même si on ne souhaite pas modifier ce dernier. Cela rend la modification des informations plus lourde que nécessaire.

Pour résoudre ce problème, il faudrait mettre en place 2 formulaires séparés : un formulaire permettant de modifier le mot de passe et un formulaire permettant de modifier le reste des informations de l'utilisateur.

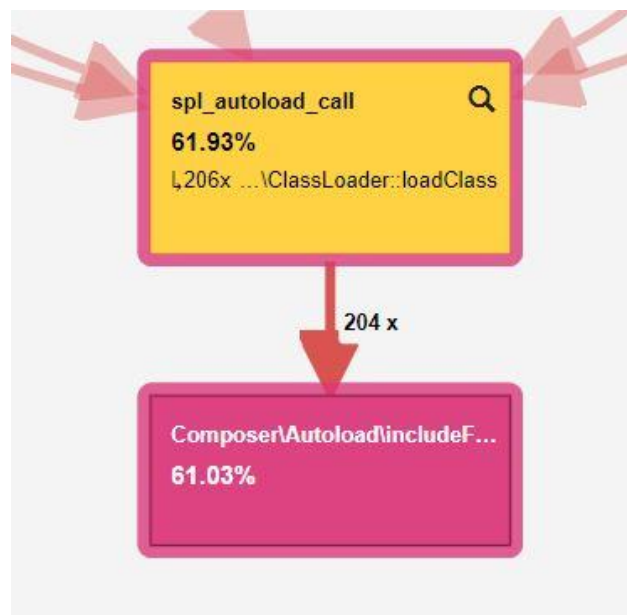
## Performance

### Le poids de l'autoloader de composer

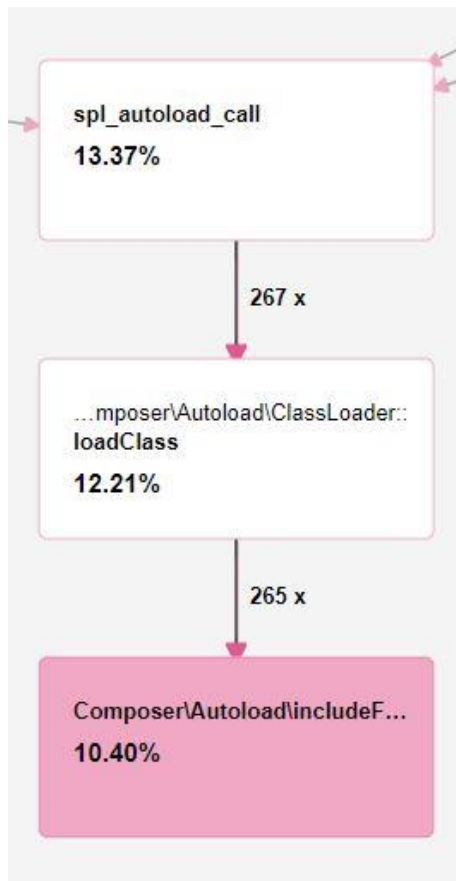
Pour optimiser les performances de l'autoloader vous devez utiliser cette commande :

**composer dump-autoload --optimize --no-dev --classmap-authoritative**

Malgré l'optimisation de l'autoloader de composer pour l'environnement de production, l'autoloader de composer mobilise autour de 60% du temps d'exécution sur l'ensemble des pages testées.



Pour limiter significativement le poids de l'autoloader dans le temps d'exécution il faut activer l'**Opcache** qui fait passer le poids de l'autoloader sous la barre des 20%.



De plus, l'**Opcache** diminue significativement le temps d'exécution des différentes actions on se retrouve alors avec des temps d'exécution oscillent selon la page entre **9ms et 60ms**.

### Minifier les feuilles de style et les fichiers JS

La feuille de style **shop-homepage.css** est utilisé par les pages dans une version non minifié, étant donné la petite taille de ce fichier l'impact sur les performances est faible mais dans le cadre de développement futur, il serait préférable d'utiliser une version minifié de ces fichiers.

Pour afficher ces fichiers en version minifié tout en préservant une version non minifié pour les développements futur, une solution possible est d'utiliser le bundle **Assetic**. Ce bundle peut permettre d'utiliser des versions minifié de nos fichiers sur nos pages, de fusionner nos fichiers en un seul pour optimiser le chargement de la page et il permet enfin d'appliquer des filtres aux différentes images pour les optimiser.

**Attention :** Si vous optez pour la version 4 de Symfony vous ne pourrez pas utiliser Assetic, il faudra alors se tourner vers **webpack** pour gérer les assets.

### Utiliser un système de compression

Le site web utilise des images, des feuilles de style et des scripts Javascript. Sans une configuration particulière leur temps de chargement n'est pas optimal et plus la taille de ces fichiers sera importante plus cela pourra peser sur le temps de chargement de la page.

Pour optimiser le temps de chargement des assets, il serait judicieux d'utiliser une compression **Deflate** ou **Gzip**. La mise en place de ces systèmes permettra la compression des fichiers par le serveur lors de leur envoi et réduira ainsi le temps de chargement de la page.