# TODO & CO

Audit de qualité du code & performance de l'application

# SOMMAIRE

# 1

# Contexte du projet

L'application TodoList appartient à la jeune startup ToDO & Co. L'application a dû être développée à toute vitesse pour présenter le concept à de potentiels investisseurs. Suite à la présentation, l'entreprise a réussi à lever des fonds pour le développement de l'application et son expansion.

Le but de cet audit est de faire état des lieux de la dette technique de l'application. Il portera sur les deux axes suivants : la qualité de code et la performance.

Dans un second temps nous parlerons de ce qui a été mis en place pour répondre aux problématiques qui ont émergées de cet audit.

# 2

# État de l’application

# Front-end

Plusieurs problèmes ont été relevés quant à l'implémentation du front-end de l'application. Point par point nous allons analyser ce qui pourrait être amélioré et les correctifs qui pourraient être appliqués

### Les librairies utilisées

Jquery est manquant des fichiers JS, du coup bootstrap.min.js ne fonctionne pas correctement. Une erreur apparaît dans la console.

Bootstrap est en version 3.3.7, qui est sorti en 2016. La dernière version est la 5.3.1, une mise à jour serait bienvenue.

Globalement, l'utilisation de Bootstrap et de Jquery est à remettre en cause pour ce projet vu sa taille, beaucoup de code n'est pas utilisé. Cela améliorerait la vitesse de chargement.

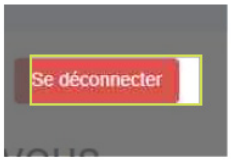
L'application utilise Glyphicons pour seulement 2 icônes, utiliser directement 2 svgs serait plus performant.

## Header

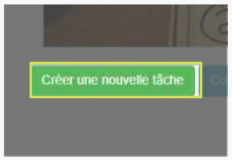
Le header n'est pas utile en l'état. Il serait intéressant d'y ajouter une navigation.

## Accéssibilité

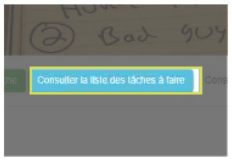
L'audit réalisé par Lighthouse fait ressortir que es textes des boutons n'ont pas assez de contraste avec la couleur de fond.




```
a.pull-right.btn.btn-danger
```




```
a.btn.btn-success
```



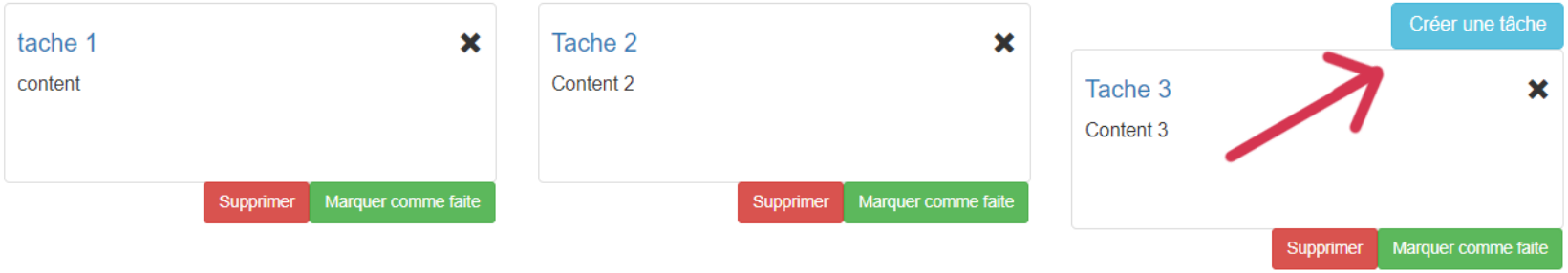

```
a.btn.btn-info
```

## Alignements

La grille Bootstrap n'est pas très bien utilisée. Une 'row' doit toujours être suivie d'une 'col' pour que les marges soient correctes.
Une 'row' ne peut pas être enfant d'une 'col' ni d'une 'row'.
L'ordre correct est le suivant : container > row > col.

## Calls to actions

Sur la page listant les tâches, le bouton pour créer une tâche n'est pas bien placé. Il se trouve au milieu des tâches qui ont déjà été créées et n'est pas assez visible.




Certains boutons ne fonctionnement pas :

- 'Consulter la liste des tâches terminées' sur l'accueil
- 'To Do List App' dans le header ne redirige vers rien, en général lorsque l'on clique sur le logo d'une application on voudrait être redirigé vers l'accueil.
- l'icône 'X' pour supprimer une tâche.

Quelques problèmes se posent sur le styles des buttons :

- 5 types de boutons sont utilisés, c'est beaucoup trop.
- Le bouton 'Consulter la liste des tâches terminées' est secondaire alors qu'il représente le même type d'action que les autres.

## Navigation

La navigation est compliquée :

- On a pas de menu pour accéder directement aux fonctionnalités.
- Quand on a quitté l'accueil, aucun bouton ne nous permet d'y retourner.
- Les CTAs ne sont pas toujours bien placés.

## Responsive

Aucune marge sur les bords en mobile dû à la mauvaise utilisation de la grille Bootstrap.

Mauvais redimensionnement du texte.

Le bouton pour créer une tâche disparaît. (/tasks)

# Back-end

## Versions des dépendances

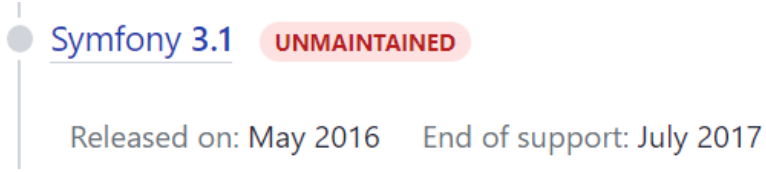
```
"require": {
    "php": ">=5.5.9",
    "symfony/symfony": "3.1.*",
```

### PHP

Depuis le 1 janier 2019, PHP 5.5 ne reçoit plus aucun correctif de sécurité. Il est nécessaire de passer à une version plus récente. La dernière version LTS de php est la 8.1.7.

### Symfony

D'après la documentation du symfony, la version 3.1 n'est plus maintenue depuis Juillet 2017. La dernière version stable à utiliser est la 6.1 et selon la roadmap de Symfony elle sera supportée jusqu'en 2023.



## Pourquoi mettre à jour les dépendances ?

Plusieurs avantages :

- **Sécurité** : La raison la plus évidente de la mise à jour est la sécurité. Les versions les plus récentes contiennents les derniers correctifs et seront maintenues plus longtemps.
- **Performance** : Les nouvelles versions sont généralement plus rapides que les précédentes, c'est notamment le cas pour PHP.
- **Fonctionnalités** : Chaque nouvelle versions met en place de nouvelles fonctionnalités que vos développeurs vont pouvoir utiliser, s'en priver c'est potentiellement perdre du temps à implémenter votre propre logique métier.

Plus on tarde à faire des migrations, plus le coût s'envole. Ne pas faire les mises à jour c'est une prise de risque. Si votre application subit une attaque, la charge de travail pour une remise en service et la potentielle perte de données peuvent être énormes.

## Principes Solid

L'application est relativement petite, donc aucune erreur majeure n'est à déplorer. Nous pouvons tout de même mettre en avant certains principes solides qui n'ont pas été respectés :

- **Single Responsibility Principle (SRP)** : Le traitement des formulaires et l'ajout ou la modification des données ne devraient pas se trouver dans les contrôleurs. Séparer la logique métier serait plus pertinent.

- **Dependency Inversion Principle (DIP)** : Il n'y a pas d'injection de dépendances, donc par définition le principe n'est pas respecté.
- (UserController l.33).

## Tests automatisés

Aucun test automatisé n'a été implémenté pour vérifier le bon fonctionnement de l'application lors des changements futurs. En faire permet à l'équipe de développement de découvrir et corriger des bugs qui par la suite deviendraient plus couteux et difficiles à résoudre à la fin du projet.

# Analyse du code

Le code de l'application a été analysé par plusieurs outils, voici ce qu'il en ressort :

- **PHP STAN (level 9)** :
  Les retours de fonction ne sont pas bien typés.
  Les classes Entity ne sont pas typées.
- **PHP MD** :
  Cleancode - 'use Datetime' à la place de '\Datetime'.
  Naming - Éviter les noms de variable courts comme $em.
  Unused - SecurityController->loginAction() : $request unused.
- **PHP CPD** : Pas de duplication de code détecté.
- **Composer** : Pas de propriété 'description'.
- **Librairies** :
  Symfony/swiftmailer-bundle - non utilisée
  doctrine/doctrine-cache-bundle - abandonnée
  sensio/distribution-bundle - abandonnée
- **Doctrine** : Le mapping des fichiers est correct.
- **Validation** : Le mot de passe n'est pas obligatoire à la création d'un compte.

# 3

# Performance

# Analyses Blackfire

| Requêtes | Ancienne version | | Nouvelle version | | Performances optimisées | |
|---|---|---|---|---|---|---|
| login_check | 712 ms | 26 mb | 727 ms | 25 mb | 512 ms | 13.5 mb |
| logout | 275 ms | 22.7mb | 276 ms | 23 mb | 77 ms | 9.39 mb |
| login | 295 ms | 23.7mb | 300 ms | 25 mb | 107 ms | 13.9 mb |
| homepage | 360 ms | 26 mb | 351 ms | 26 mb | 142 ms | 16 mb |
| task_list | 354 ms | 26.7mb | 433 ms | 27.1 mb | 148 ms | 16.2 mb |
| GET task_create | 442 ms | 38 mb | 370 ms | 30 mb | 159 ms | 18.7 mb |
| POST task_create | 606ms | 40mb | 541 ms | 31.9mb | 185 ms | 18.5 mb |

| Requêtes | Ancienne version | | Nouvelle version | | Performances optimisées | |
|---|---|---|---|---|---|---|
| GET task_edit | 442 ms | 38.7mb | 399 ms | 30 mb | 163 ms | 18.7 mb |
| POST task_edit | 485ms | 41 mb | 497 ms | 31.8 mb | 167 ms | 18.4 mb |
| task_toggle | 330 ms | 26 mb | 332 ms | 26.1 mb | 140 ms | 14.3 mb |
| task_delete | 1060ms | 39 mb | 270 ms | 23.3mb | 71 ms | 9.42 mb |
| user_list | 879 ms | 27 mb | 319 ms | 26 mb | 134ms | 16.1 mb |
| GET user_create | 894ms | 38 mb | 360 ms | 30.7mb | 166 ms | 19.2 mb |
| POST user_create | 1230ms | 41 mb | 977 ms | 25 mb | 574 ms | 19.3 mb |
| GET user_edit | 887 ms | 38 mb | 392 ms | 30 mb | 166 ms | 19 mb |
| POST user_edit | 1560ms | 41 mb | 534 ms | 32 mb | 178 ms | 18.9 mb |

# Commentaires sur les analyses

Blackfire nous donne le temps de réponse moyen en millisecondes de nos différentes requêtes ainsi que la mémoire utilisée. Il nous est possible de voir les différentes fonctions exécutées par notre script et voir lesquelles sont les plus gourmandes.

### Autoloader

Lors de l'analyse on constate un nombre important d'appels aux fonctions «*file_exist*» et «*class_exist*» ce qui correspond à l'autoloader.
Pour optimiser cette partie, il faut lancer la commande «*composer dump-autoload --no-dev --classmap-authoritative*» :

- **composer dump-autoload** : Génère un nouveau mappage des classes utilisées dans notre application.
- **--no-dev** : Exclut les classes qui sont seulement nécessaires en environnement de développement.
- **--classmap-authoritative** : Créé un mappage des classes utilisées dans votre application et empêche Composer d'analyser les classes qui ne se trouvent pas dans le mappage des classes.

## OPcache

Lorsqu'un script PHP est exécuté, il est compilé en opcode, code compréhensible par la machine. OPcache stocke ce code en mémoire lors de la première exécution, pour être réutilisé par la suite. Paramétrer correctement l'OPcache avec les recommandations de Symfony permet d'améliorer significativement les performances.

## PHP realpath Cache

Lorsqu'un chemin de fichier relatif est transformé en chemin absolu, PHP met en cache le résultat pour améliorer les performances. Une application Symfony devrait utiliser au moins ces valeurs:
realpath_cache_size=4096K
realpath_cache_ttl=600

## Conclusion

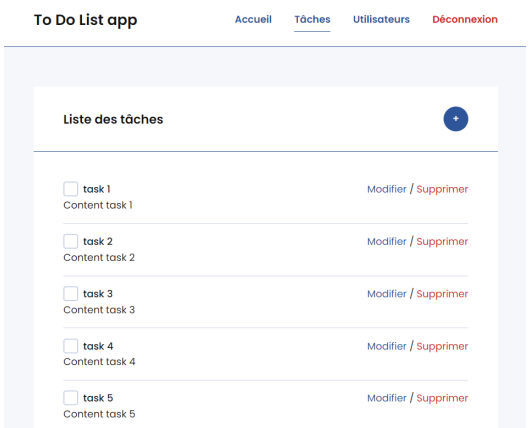
Avant optimisation, on ne remarque par forcément un gain de performance entre l'ancienne version et la nouvelle version, mais après avoir effectué les optimisations détaillées ci-dessus, on remarque un gain de plus de 50% pour la plupart des requêtes. Plus d'infos sur l'optimisation des performances dans la documentation de Symfony.

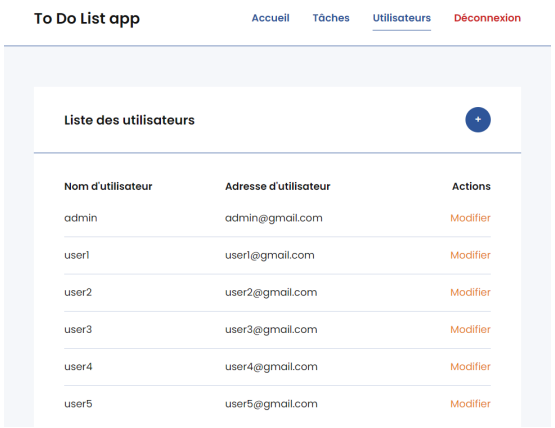
# 4

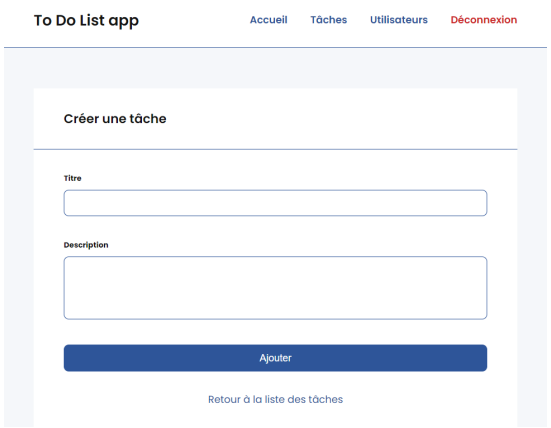
# Correctifs

# Refonte du Front-end

La partie front utilisait des librairies qui n'étaient pas forcément essentielles pour un projet de cette envergure, et plusieurs problèmes d'ergonomie avaient été relevés. Il a donc été décidé de refaire une refonte de toute cette partie.
L'application n'utilise donc plus Jquery ni Bootstrap mais du vanilla CSS et JS pour de meilleures performances. L'UI et l'UX ont aussi été corrigés pour une utilisation plus simple de l'application par ses utilisateurs.



# Refonte du Back-end

Le projet avait accumulé de la dette technique au fil du temps, notamment du côté des versions des packages utilisés et de la version de php utilisée. Il a été décidé de repartir de zéro en utilisant les dernières versions disponibles à ce jour :

- **php** : 8.0
- **symfony** : 6.0
- **doctrine/doctrine-bundle** : 2.5

# Implémentation des fonctionnalités

Il avait été demandé que de nouvelles fonctionnalités soient implémentées dans l'application, voici ce qui a changé :

- Seuls les administrateurs peuvent accéder aux pages de gestion des utilisateurs.
- Les tâches ne peuvent être supprimées que par les utilisateurs ayant créé les tâches en question.
- Les tâches rattachées à aucun utilisateur ne peuvent être supprimées que par un administrateur.
- Lors de la création ou de la modification d'un utilisateur, il est possible de lui attribuer un rôle : ROLE_USER, ROLE_ADMIN.
- Des tests automatisés ont été implémentés avec PhpUnit. Le rapport de couverture est disponible ici.