OPENCLASSROOMS

AUDIT DE QUALITE DE CODE ET DE PERFORMANCE

TODO & CO

22/06/2022

Rédigé par Siaka MANSALY

TABLE DES MATIERES

A. Contexte 1

B. Projet initial - Etat des lieux 2

1) Environnement de l'application 2
2) Démarrage du projet 2
3) Analyse du code 3
   - Framework 3
   - Base de données 4
   - Dépendances 4
   - Fichiers de code, configuration et Views 5
   - Analyse complète 6
4) Analyse du Frontend 7
   - Analyse Lighthouse 7
   - Analyse W3C 7
   - Affichage et Navigabilité 8
   - Sécurité 8
5) Analyse des performances 9

C. Recommandations 10

D. Projet modifié - Récapitulatif 11

1) Démarrage du projet 11
2) Mise à jour de l'application 11
   - Migration du Framework 11
   - Correctifs liés à la migration 12
3) Implémentation de nouvelles fonctionnalités 14
   - Tâches 14
   - Utilisateurs 15
4) Corrections des anomalies 15
   - Tâches 15
   - Utilisateurs 16
   - Frontend 16
5) Implémentations des tests automatisés 17
6) Analyse du code 18
   - Dépendances 18
   - Base de données 18
   - Fichiers de code, views et de configuration 18
   - Analyse complète 19
7) Analyse du Frontend 20
8) Analyse des performances 20
   - Récapitulatif des optimisations 20
   - Comparatif général des performances 21
   - Plus de détails 22

E. Conclusions générales 23

A. CONTEXTE

Auditeur : Siaka MANSALY

Date de l'audit : 24/06/2022

La startup ToDo & Co met à disposition une **application** permettant à ses utilisateurs de **gérer leurs tâches quotidiennes**.

L'application a été développée avec le **minimum viable** (MVP) sous le **Framework PHP Symfony** (version 3.1.10) pour montrer à de potentiels investisseurs la viabilité du concept.

Une fois le projet financé, il m'a été demandé de :

- **Améliorer** la qualité de l'application et **réduire la dette technique** de l'application
- **Implémenter** de nouvelles **fonctionnalités**
- **Corriger** quelques anomalies
- Implémenter des **tests automatisés**

Je vous présenterai dans un premier temps un **état des lieux** de l'application avant modification.

Outils de référence utilisés pour cet audit :

- Blackfire : https://www.blackfire.io/
- Codacy : https://www.codacy.com/
- insolita/unused-scanner : https://packagist.org/packages/insolita/unused-scanner
- Lighthouse : https://developer.chrome.com/docs/lighthouse/overview/
- PHP Coding Standards Fixer : https://cs.symfony.com/
- PHP-FIG : https://www.php-fig.org/psr/
- PHPStan : https://phpstan.org/user-guide/getting-started
- Symfony : https://symfony.com/
- Web Developper : https://chrispederick.com/work/web-developer/
- W3C : https://validator.w3.org/

B. PROJET INITIAL - ETAT DES LIEUX

1) Environnement de l'application

L'application a été développée sous le **Framework PHP Symfony version 3.1.10**.

Elle dispose des **dépendances** suivantes :

```
"require": {
    "php": ">=5.5.9",
    "symfony/symfony": "3.1.*",
    "doctrine/orm": "^2.5",
    "doctrine/doctrine-bundle": "^1.6",
    "doctrine/doctrine-cache-bundle": "^1.2",
    "symfony/swiftmailer-bundle": "^2.3",
    "symfony/monolog-bundle": "^2.8",
    "symfony/polyfill-apcu": "^1.0",
    "sensio/distribution-bundle": "^5.0",
    "sensio/framework-extra-bundle": "^3.0.2",
    "incenteev/composer-parameter-handler": "^2.0"
},
"require-dev": {
    "sensio/generator-bundle": "^3.0",
    "symfony/phpunit-bridge": "^3.0"
},
```

2) Démarrage du projet

J'ai tout d'abord effectué une **mise à jour mineure** de Symfony vers la version 3.4.49 afin de pouvoir profiter des fonctionnalités de **PHP 7** dans le cadre de mon analyse.

Prérequis :

- PHP : >= 7.1 : https://www.php.net/downloads.php
- Composer <= 2.2 : https://getcomposer.org/download/
- Symfony CLI : https://symfony.com/download

Etapes d'installation :

1. Copie de l'ensemble du projet en local :
   https://github.com/siakamansaly/Audit-and-Improve-Symfony-App/tree/feature/upgrade-to-symfony-3.4.49

2. Installation des dépendances et configuration des variables d'environnement :
   ```
   composer install
   ```

3. Création de la base de données
   ```
   php bin/console doctrine:database:create
   php bin/console doctrine:schema:update --force
   ```

4. Démarrage de l'application
   ```
   php bin/console server:run
   ```

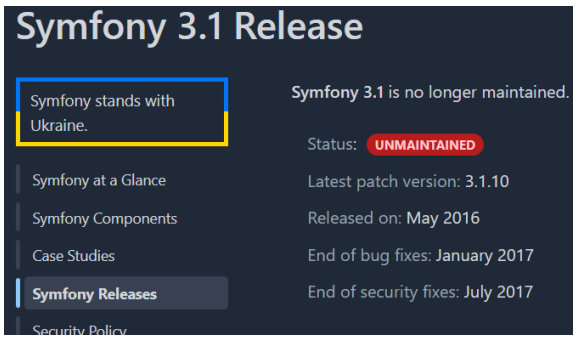
3) Analyse du code

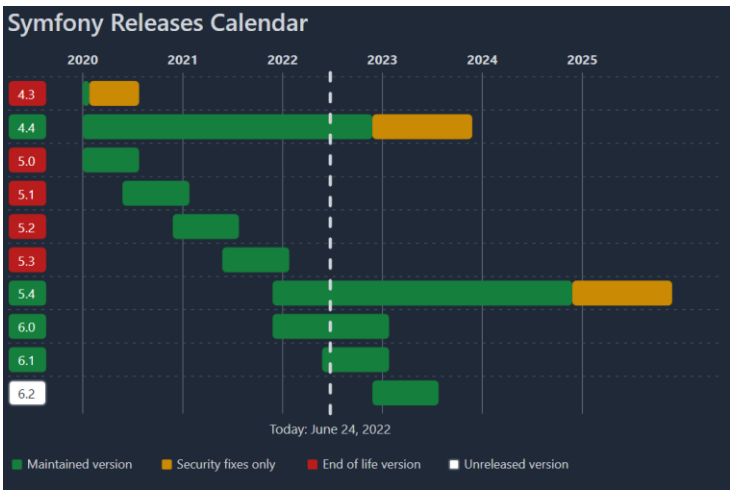
➢ Framework

L'application ToDo & Co développée sous le Framework **Symfony 3.1** n'est **plus maintenue** depuis 2017 et nous expose à des **problèmes de sécurité**. La version 3.4.49 n'est plus maintenue également.

Une **mise à jour vers une version LTS** (Long-Term Support) serait l'idéal. A ce jour, la version 5.4.9 est stable et dispose de la plus longue LTS avec un support garanti d'au moins 3 ans.

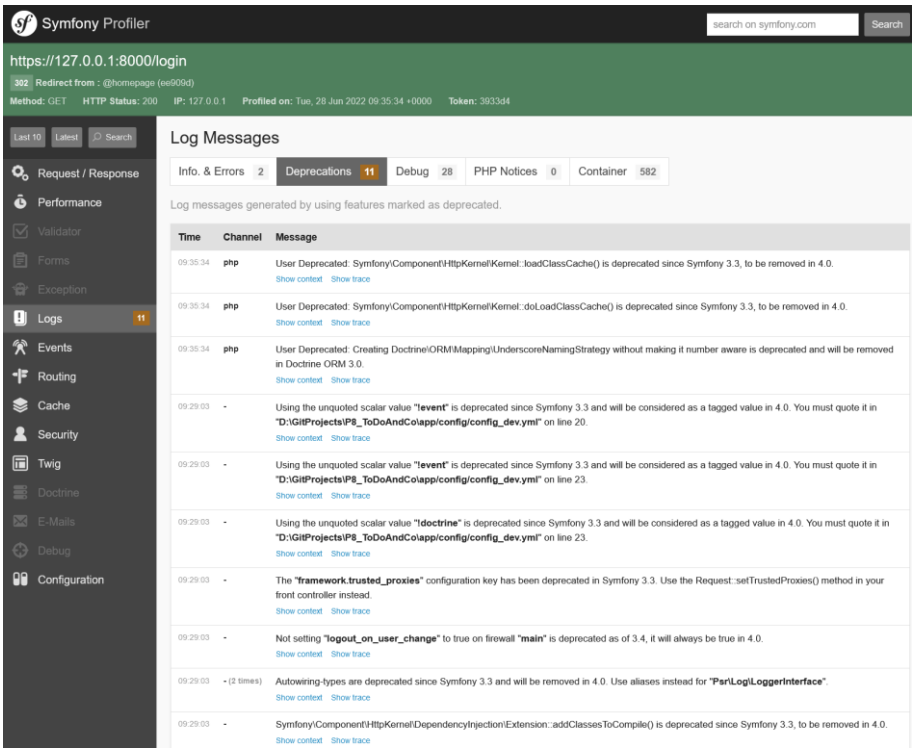
Sources : https://symfony.com/releases





J'ai également consulté le **Profiler** de Symfony qui confirme l'obsolescence de la version du Framework par rapport à ses dépendances.

Exemple ci-dessous sur la page « Login » :



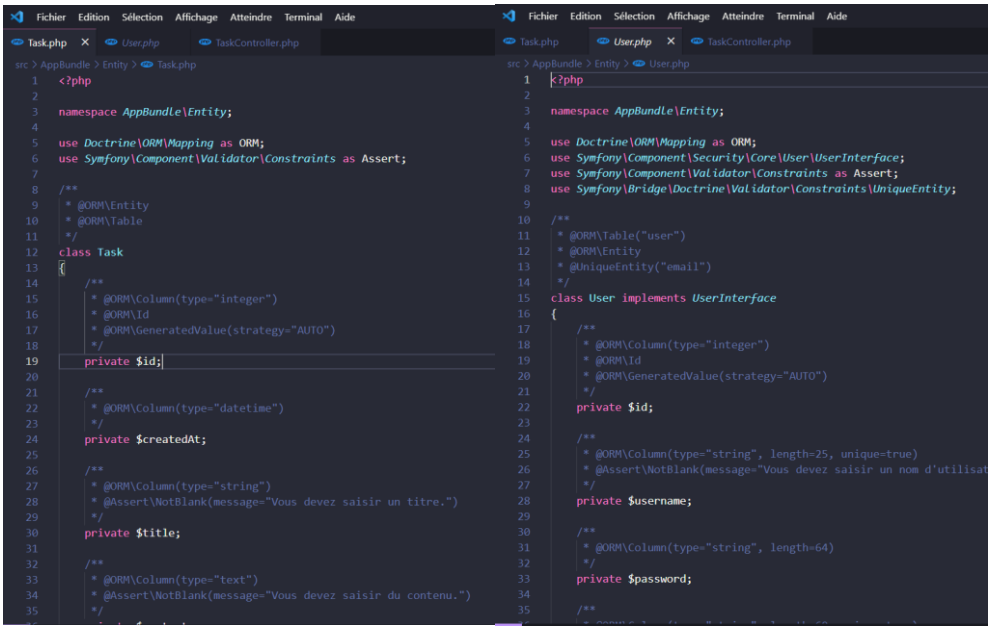
➢ Base de données

La commande interne « **doctrine:schema:validate** » m'a permis de **vérifier la base de données**.

Le mappage des fichiers doctrine est bien fonctionnel et la base de données se synchronise bien avec les fichiers doctrine.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console doctrine:schema:validate

Mapping
-------

[OK] The mapping files are correct.

Database
--------

[OK] The database schema is in sync with the mapping files.
```

Il y a 2 entités « **User** » et « **Tasks** ». Cependant, il n'existe **aucune relation** entre celles-ci actuellement.



```php
<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * @ORM\Entity
 * @ORM\Table
 */
class Task
{
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(type="datetime")
     */
    private $createdAt;

    /**
     * @ORM\Column(type="string")
     * @Assert\NotBlank(message="Vous devez saisir un titre.")
     */
    private $title;

    /**
     * @ORM\Column(type="text")
     * @Assert\NotBlank(message="Vous devez saisir du contenu.")
     */
```

```php
<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/**
 * @ORM\Table("user")
 * @ORM\Entity
 * @UniqueEntity("email")
 */
class User implements UserInterface
{
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=25, unique=true)
     * @Assert\NotBlank(message="Vous devez saisir un nom d'utilisat
     */
    private $username;

    /**
     * @ORM\Column(type="string", length=64)
     */
    private $password;
```

➢ Dépendances

La commande interne « **composer valid** » de Composer nous a permis d'analyser le fichier des **dépendances** de l'application.

```
PS D:\GitProjects\P8_ToDoAndCo> composer valid
./composer.json is valid for simple usage with Composer but has
strict errors that make it unable to be published as a package
See https://getcomposer.org/doc/04-schema.md for details on the schema
# Publish errors
- description : The property description is required
# General warnings
- Defining autoload.psr-4 with an empty namespace prefix is a bad idea for performance
```

Il y a 2 erreurs dans le fichier « **composer.json** » :

Erreur	Description
La description de l'application est manquante.	Cela n'a pas d'impact sur les performances, mais il s'agit d'une bonne pratique de développement.
Le Namespace de l'autoloader est également manquant.	Cela peut avoir un impact au niveau des performances de l'application.

➢ Fichiers de code, configuration et Views

Les **URL** sont bien paramétrées via des annotations et respectent les bonnes pratiques.

Le code PHP n'est pas **documenté**. Une **documentation** via des annotations **PhpDoc** serait un plus pour la suite du projet.

PHPStan

Une analyse poussée (Niveau 9) des fichiers du dossier « **src** » a été effectuée grâce à l'outil « **PHPStan** ». Il a pour rôle de détecter les bogues au niveau du code PHP.

Celle-ci nous a révélé principalement un **manque de spécification de type** au niveau des classes (56 bogues).

Il serait judicieux de corriger ces manquements afin d'éviter des résultats inattendus à l'avenir.

Commande PHPStan : « vendor/bin/phpstan analyse »

```
------ ------------------------------------------------------------------
 Line   src\AppBundle\Form\TaskType.php
------ ------------------------------------------------------------------
 11     Method AppBundle\Form\TaskType::buildForm() has no return type specified.
------ ------------------------------------------------------------------

------ ------------------------------------------------------------------
 Line   src\AppBundle\Form\UserType.php
------ ------------------------------------------------------------------
 14     Method AppBundle\Form\UserType::buildForm() has no return type specified.
------ ------------------------------------------------------------------

------ ------------------------------------------------------------------
 Line   tests\AppBundle\Controller\DefaultControllerTest.php
------ ------------------------------------------------------------------
 9      Method Tests\AppBundle\Controller\DefaultControllerTest::testIndex() has no return type specified.
 15     Call to an undefined method Tests\AppBundle\Controller\DefaultControllerTest::assertEquals().
 15     Cannot call method getStatusCode() on Symfony\Component\HttpFoundation\Response|null.
 16     Call to an undefined method Tests\AppBundle\Controller\DefaultControllerTest::assertContains().
------ ------------------------------------------------------------------

[ERROR] Found 56 errors
```

Commande PHP-CS-Fixer :
« php tools/php-cs-fixer/vendor/bin/php-cs-fixer list-files »

```
PS D:\GitProjects\P8_ToDoAndCo> php tools/php-cs-fixer/vendor/bin/php-cs-fixer list-files
".\app\AppCache.php"
".\app\AppKernel.php"
".\app\autoload.php"
".\src\AppBundle\AppBundle.php"
".\src\AppBundle\Controller\DefaultController.php"
".\src\AppBundle\Controller\SecurityController.php"
".\src\AppBundle\Controller\TaskController.php"
".\src\AppBundle\Controller\UserController.php"
".\src\AppBundle\Entity\Task.php"
".\src\AppBundle\Entity\User.php"
".\src\AppBundle\Form\TaskType.php"
".\src\AppBundle\Form\UserType.php"
".\tests\AppBundle\Controller\DefaultControllerTest.php"
".\web\app.php"
".\web\app_dev.php"
".\web\config.php"
```

PHP CS Fixer

L'outil PHP Coding Standards Fixer m'a permis d'identifier les fichiers comportant des **problèmes de normes de codage** telles que définies dans les documents PSR-1 et PSR-2 et bien d'autres.

unused-scanner

L'analyse avec l'outil « **unused-scanner** » m'a, quant à lui, répertorié les **dépendances inutilisées** dans l'application.

Dépendances inutilisées :

- symfony/polyfill-apcu
- incenteev/composer-parameter-handler
- doctrine/doctrine-cache-bundle

Il faudrait penser à supprimer celles-ci afin d'alléger le projet.

```
PS D:\GitProjects\P8_ToDoAndCo> unused_scanner scanner_config.php
 - config prepared
 - search patterns prepared

 - Scan D:\GitProjects\P8_ToDoAndCo\app
100%[============================================================>]
 - Scan D:\GitProjects\P8_ToDoAndCo\src
100%[============================================================>]
 - Scan D:\GitProjects\P8_ToDoAndCo\tests
100%[============================================================>]
 - Scan D:\GitProjects\P8_ToDoAndCo\web
100%[============================================================>]
 - Scan D:\GitProjects\P8_ToDoAndCo\bin

 - Scan  additional files
100%[============================================================>]
Unused dependencies found!
 -symfony/polyfill-apcu
 -incenteev/composer-parameter-handler
 -doctrine/doctrine-cache-bundle
```

Une **analyse des fichiers de configuration YAML** a été effectuée. Une erreur est ressortie dans le fichier « config_dev.yml ». Il s'agit d'une **dépréciation** par rapport à la version du Framework Symfony. Cette erreur n'est pas pénalisante pour l'environnement de production car elle se situe dans l'environnement de développement uniquement.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console lint:yaml app/config/

 ERROR  in app/config\config_dev.yml
 >> Using the unquoted scalar value "!event" is deprecated since Symfony 3.3 and will be considered as a tagged value in 4.0. You must quote it on line 2
0 at line 20 (near "channels: [!event]").

[WARNING] 8 YAML files have valid syntax and 1 contain errors.
```

Les **fichiers Twig** permettant de générer l'affichage de l'application ont également été analysés. L'ensemble des fichiers Twig ont une syntaxe valide.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console lint:twig app/Resources/

[OK] All 9 Twig files contain valid syntax.
```

L'application utilise actuellement la version 3.3.7 de **Bootstrap** et une ancienne version de **Jquery**. Ces versions sont vulnérables car certains scripts tiers peuvent présenter des failles de sécurité connues, faciles à identifier et à exploiter par des pirates informatiques.

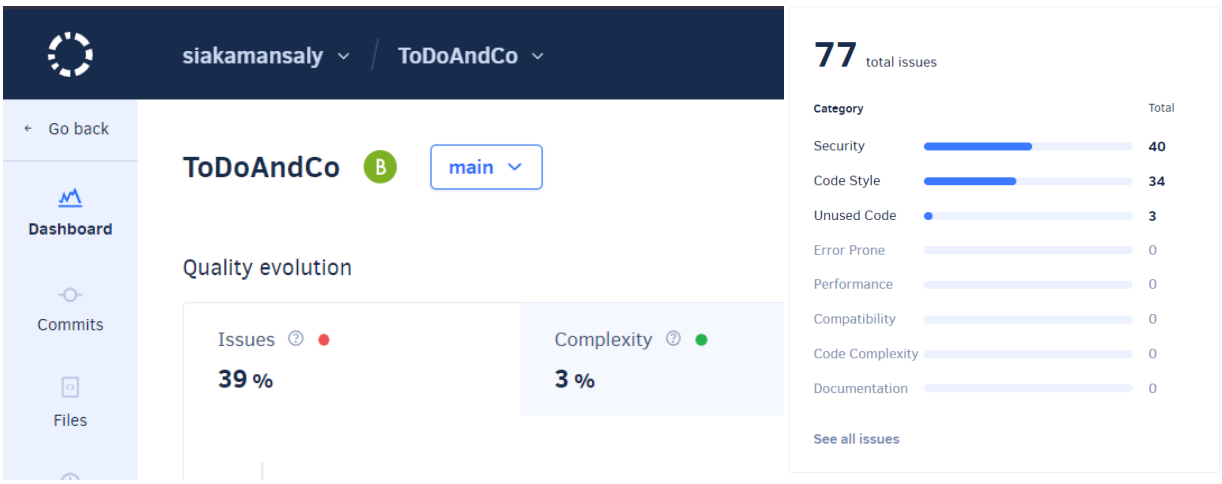
Une mise à jour de Bootstrap et Jquery est nécessaire.

➢ **Analyse complète**

Codacy permet d'effectuer une révision du code et permet la surveillance de la qualité du code.
Le code de l'application ToDo & Co a été analysé entièrement sous **Codacy** et dispose d'une bonne notation (B).

Cependant, on relève que 40 **problèmes de sécurité** ont été détectés dont 17 problèmes critiques.

Lien : https://app.codacy.com/gh/siakamansaly/Audit-and-Improve-Symfony-App/dashboard



Exemple d'erreur critique :

Le fichier « app_dev.php » nous montre l'utilisation au sein de l'application de variables superglobales qui sont fortement déconseillés.

Il faudrait penser à encapsuler celles-ci en enveloppant les superglobales dans une classe ou en utilisant une dépendance de Symfony comme « HttpFoundation » par exemple.

4) Analyse du Frontend

➢ Analyse Lighthouse

L'analyse grâce à l'outil Google **Lighthouse** est bonne cependant certaines amélioration et corrections sont nécessaires.

Exemple Page d'accueil

Erreurs / Améliorations relevées (sur la page d'accueil, la page créer un utilisateur et la page créer une tâche) :

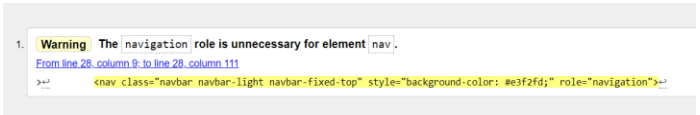
1. Réduisez les ressources CSS inutilisées
2. Les éléments d'image ne possèdent pas de width ni de height explicites
3. Diffusez des éléments statiques grâce à des règles de cache efficaces
4. Les couleurs d'arrière-plan et de premier plan ne sont pas suffisamment contrastées
5. Les erreurs de navigateur ont été enregistrées dans la console (Bootstrap et Jquery)
6. Le document ne contient pas d'attribut "meta description". Le texte de la description est vide.
7. Les liens ne peuvent pas être explorés (« Consulter la liste des tâches terminées »)

Il serait judicieux de corriger les points 5, 6 et 7 en priorité afin de ne pas perturber l'expérience utilisateur.

➢ Analyse W3C

L'analyse **W3C** du code **HTML** est excellente. Une seule recommandation a été remonté.

Il est préconisé de supprimer l'attribut rôle de la balise <nav> car ce n'est pas pertinent.



L'analyse **W3C** du code **CSS** est valide.

Il a 6 erreurs remontées avec notamment des propriétés CSS inexistantes ou erronées.

➢ Affichage et Navigabilité

Une analyse a été effectuée à l'aide du **Responsive Layout** de **Web Developer**.

L'application est bien adaptée à différents écrans.

Cependant, il est **compliqué de naviguer sur l'application**. Pour revenir à la page d'accueil, on est obligé d'utiliser les fonctions de notre navigateur internet ou taper directement l'URL dans la barre de navigation.

Dans la page « /tasks », il serait pertinent de **rajouter un filtre** permettant de différencier les tâches à faire et les tâches terminées.

Un **menu de navigation** général serait idéal pour l'ensemble de l'application.

➢ Sécurité

En navigant sur l'application, on constate que les différentes **pages** de l'application sont accessibles **sans authentification**. Notamment, les pages permettant la gestion des utilisateurs ainsi que la gestion des tâches.

Les **formulaires** ne contiennent pas de jetons pour être protégé contre les attaques **CSRF**.

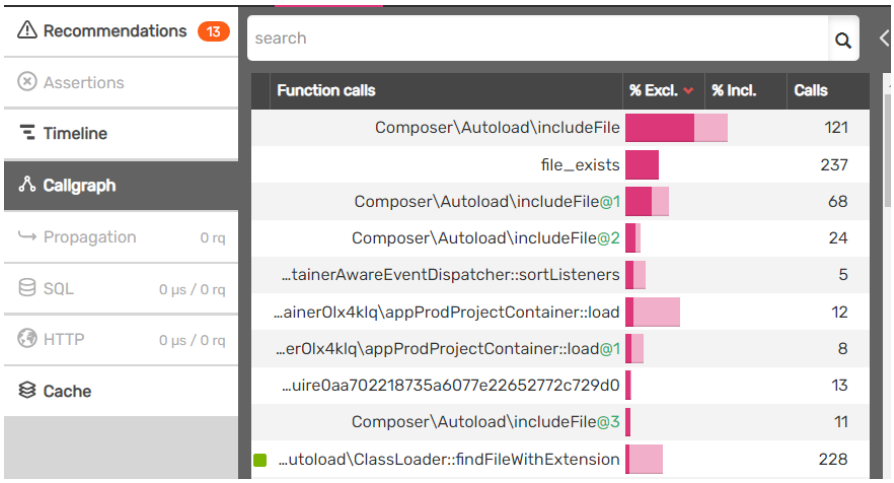
Il serait judicieux de **gérer les droits d'accès** aux pages du site et **sécurisé les formulaires**.

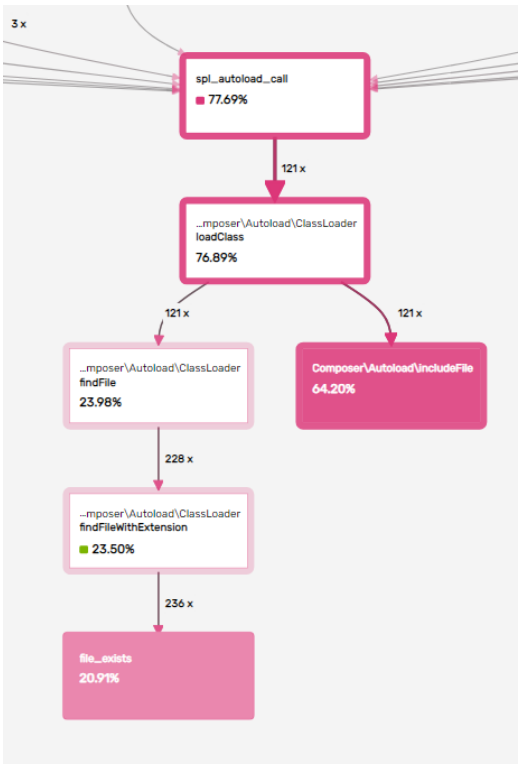
5) Analyse des performances

Toutes les routes de l'application ont été profilés sur l'outil **Blackfire.io** afin de tester les performances.

Vous trouverez ci-dessous les métriques pour chacune des pages :

URL (Routes)	Wall Time (ms)	I/0 Wait (ms)	CPU Time (ms)	Memory (MB)
/	254	203	51.3	12.2
/tasks	262	204	58	12.3
/tasks/create	315	245	69.9	15.8
/tasks/{id}/edit	320	246	74.7	15.8
/login	171	134	37.1	7.94
/users	271	214	57.3	12.2
/users/create	292	228	64.3	15.7
/users/{id}/edit	339	258	81.7	15.8





On constate que certaines ressources sont appelées plusieurs fois et pourraient être optimisées pour améliorer le chargement des pages.

D'ailleurs, Blackfire nous a remonté plusieurs recommandations pour gagner en performance :

- Optimiser le chargement de Composer en production
- Activer la mise en cache des annotations de doctrine

C. RECOMMANDATIONS

Vous trouverez ci-dessous un récapitulatif des différentes recommandations proposées pour l'application.

N	Libellé	Description
1	**Framework**	- La version du Framework est obsolète. Une mise à jour minimum vers la version 5.4 est recommandée. Une mise à jour corrigera des problèmes de sécurité notamment.
2	**Dépendances**	- Rajouter une description et un namespace à Composer - Désinstaller les dépendances inutilisées afin d'alléger le projet
3	**Code**	- Ajouter annotations PhpDoc pour documenter le code - Spécifier les formats de sortie au niveau des méthodes afin d'éviter les bogues inattendus - Fixer le code des fichiers pour respect des bonnes pratiques PSR-1 et PSR-2
4	**Frontend**	- Corriger les erreurs du navigateur et les liens inexplorés - Rajouter une meta description pour le référencement - Corriger les propriétés CSS erronées ou non pertinentes - Mettre à jour Bootstrap et Jquery avec des versions plus récentes
5	**Performance**	- Optimiser le chargement des dépendances (Composer) - Activer la mise en cache
6	**Design et Navigation**	- Ajout d'une barre de navigation pour simplifier la navigation entre les différentes pages de l'application. - Rajouter un filtre permettant de différencier les tâches à faire et les tâches terminées. - Améliorer le design du frontend pour rendre l'application plus attrayante.

D. PROJET MODIFIE - RECAPITULATIF

1) Démarrage du projet

Prérequis :

- PHP : >= 7.4.0 : https://www.php.net/downloads.php
- Composer : https://getcomposer.org/download/
- Symfony CLI : https://symfony.com/download

Etapes d'installation :

1. Copie de l'ensemble du projet en local : https://github.com/siakamansaly/Audit-and-Improve-Symfony-App
2. Installation des dépendances :

```
composer install
```

3. Création de la base de données

```
php bin/console doctrine:database:create
php bin/console doctrine:schema:update --force
```

4. Création de la base de données de test (optionnel)

```
php bin/console doctrine:database:create --env=test
php bin/console doctrine:schema:update --force --env=test
```

5. Démarrage de l'application

```
php bin/console server:run
```

Version du projet

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console about
 -------------------- --------------------------------
  Symfony
 -------------------- --------------------------------
  Version              5.4.10
  Long-Term Support    Yes
  End of maintenance   11/2024 (in +871 days)
  End of life          11/2025 (in +1236 days)
 -------------------- --------------------------------
  Kernel
 -------------------- --------------------------------
  Type                 App\Kernel
  Environment          prod
  Debug                false
  Charset              UTF-8
  Cache directory      ./var/cache/prod (672 KiB)
  Build directory      ./var/cache/prod (672 KiB)
  Log directory        ./var/log (15.6 MiB)
 -------------------- --------------------------------
  PHP
 -------------------- --------------------------------
  Version              7.4.27
  Architecture         64 bits
  Intl locale          fr_FR
  Timezone             UTC (2022-07-13T23:21:04+00:00)
  OPcache              false
  APCu                 false
  Xdebug               true
 -------------------- --------------------------------
```

2) Mise à jour de l'application

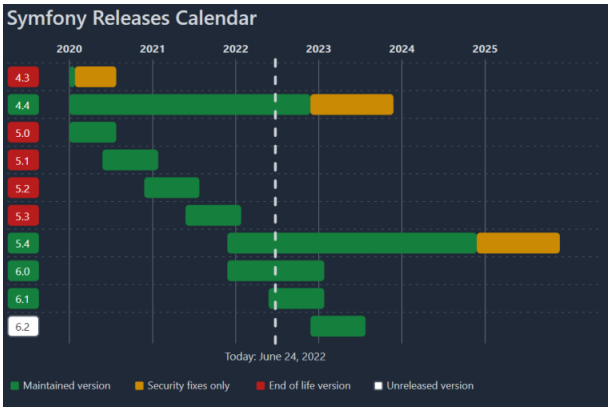
➢ Migration du Framework

Il m'a été demandé d'améliorer la qualité et de réduire la dette technique de l'application.

C'est pourquoi, ma première action a été la **mise à jour du Framework** vers une version LTS.

J'ai choisi la version 5.4 qui dispose de la maintenance logicielle et sécurité la plus longue (jusqu'en 2025 minimum).

Calendrier Releases de Symfony



Les grandes étapes de la migration ont été :

- Installation Symfony 5.4.10
- Copie des fichiers des différents dossiers (src, views, config, public)
- Installation des différentes dépendances de l'application via composer.
- Correction des erreurs post migration (Authentification, formulaires, controllers …)

➢ Correctifs liés à la migration

Un namespace « **App** » a été ajouté dans le gestionnaire des dépendances Composer. Tous les namespaces du dossier « **src** » ont été mis à jour.

Autoload du fichier « Composer.json »

```
61      "autoload": {
62          "psr-4": {
63              "App\\": "src/"
64          }
65      },
66      "autoload-dev": {
67          "psr-4": {
68              "App\\Tests\\": "tests/"
69          }
```

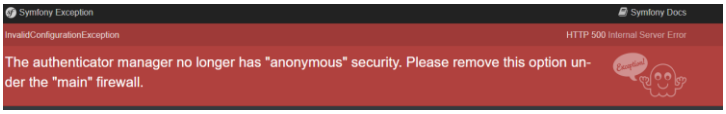
Depuis la version 5.4 de Symfony, les controllers étendent dorénavant la classe « **AbstractController** ». Tous les fichiers se situant dans le namespace « **App/Controller** » ont été modifiés.

Classe « SecurityController »

```
1   <?php
2
3   namespace App\Controller;
4
5   use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6   use Symfony\Component\HttpFoundation\Response;
7   use Symfony\Component\Routing\Annotation\Route;
8   use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;
9
10  /** ...
17  class SecurityController extends AbstractController
18  {...
45  }
```

Les **paramètres dépréciés** des fichiers de configurations ont été modifiés.

*Exemple : Suppression du paramètre « **anonymous: ~**» dans le firewall. Remplacement par le paramètre « **lazy: true** ».*



Le **système d'authentification** a été réadapté à la nouvelle version de Symfony :

- Remplacement du paramètre « **encoders** » par « **password_hashers** »

Ancienne configuration

```
security:
    encoders:
        App\Entity\User: bcrypt
```

Nouvelle configuration

```
password_hashers:
    #Symfony\Component\Security
    App\Entity\User:
        algorithm: bcrypt
```

- Remplacement des fichiers et paramètres de l'authentificateur

Ancienne configuration

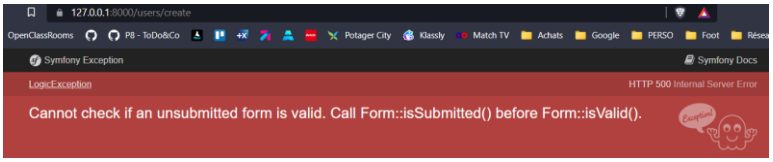
```
main:
    anonymous: ~
    pattern: ^/
    form_login:
        login_path: login
        check_path: login_check
        always_use_default_target_path:  true
        default_target_path:  /
    logout: ~
```

Nouvelle configuration

```
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        pattern: ^/
        custom_authenticator: App\Security\SecurityControllerAuthenticator
        logout:
            path: app_logout
```

Une **erreur** supplémentaire est survenue lors de l'exécution d'un **formulaire** et m'a demandé de vérifier la soumission du formulaire avant la validation. J'ai donc rajouté la **vérification de la soumission** (« **$form->isSubmitted()** ») sur tous les formulaires des controllers.

Erreur lors de la soumission d'un formulaire



Ajout de la vérification « $form->isSubmitted() »

```php
if ($form->isSubmitted() && $form->isValid()) {
    $user = $this->getUser();
    if (!$user instanceof User) {
        $user = $this->userService->userByDefault();
    }
    $task->setUser($user);
```

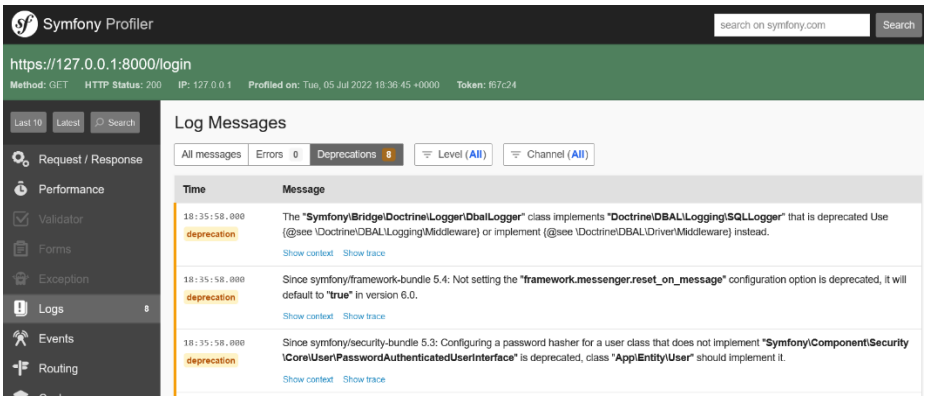
Les **formulaires** de la version 5.4 de Symfony incluent dorénavant des **jetons CSRF** par défaut.

Exemple formulaire de connexion

```html
<form method="post">
    <label for="username">Nom d'utilisateur :</label>
    <input type="text" id="username" name="_username" value>
    <label for="password">Mot de passe :</label>
    <input type="password" id="password" name="_password">
    <input type="hidden" name="_csrf_token" value="9a4aa.BHpsazs5LAhIiERObhrHzlz94
    lQnD1YUSwbNJw"> == $0
    <button class="btn btn-success" type="submit">Se connecter</button>
</form>
</div>
```

J'ai également **corrigé** sur l'ensemble des pages les **erreurs** et **dépréciations** présentes dans les logs du **Profiler** de Symfony.

Exemple Logs Profiler



3) Implémentation de nouvelles fonctionnalités

Des nouvelles **fonctionnalités** ont été **implémentées**. Vous trouverez ci-dessous les modifications apportées.

➢ Tâches

Une **tâche** ne peut être **supprimée** que par son **propriétaire**.

Création d'un voter qui vérifie si l'utilisateur actuellement connecté est bien le propriétaire de la tâche et autorise ou non l'action de suppression

```php
/**
 * Check if the user is allowed to delete a task.
 *
 * @param Task $task the task to delete
 * @param User $user the user who wants to delete the task
 *
 * @return bool true if the user is allowed to delete the task, false otherwise
 */
public function canDelete(Task $task, User $user): bool
{
    return $task->getUser() === $user;
}
```

Les **tâches** rattachées à l'utilisateur « **anonymous** » (ou sans propriétaire) pourront être **supprimées** uniquement par les utilisateurs ayant le **rôle administrateur**.

Création d'un voter qui vérifie si l'utilisateur actuellement connecté a bien le rôle administrateur et que la tâche est bien rattaché à un utilisateur anonyme (ou nulle) et autorise ou non l'action de suppression.

```php
$isAnonymous = (null === $subject->getUser()) ? true : ($subject->getUser()->isAnonymous());

if ($this->security->isGranted('ROLE_ADMIN', $user) && $isAnonymous) {
    return true;
}
```

➢ Utilisateurs

Un bouton « **Gérer les utilisateurs** » permettant d'accéder à la liste des utilisateurs a été créée.

Les **droits d'accès** aux pages de gestion des utilisateurs ont été **restreints** (uniquement accessible aux utilisateurs ayant le rôle administrateur).

Modification du rôle autorisé pour toutes les routes commençant par « /users »

```yaml
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/, roles: ROLE_USER }
```

4) Corrections des anomalies

Les **anomalies** que vous nous avez remonter ont également été **corrigées**. Vous trouverez les correctifs ci-dessous.

➢ Tâches

Une **tâche** créée est désormais automatiquement **rattachée** à **l'utilisateur** connecté.

Création d'une relation « ManyToOne » entre l'entité « User » et l'entité « Task »

```php
/**
 * @ORM\ManyToOne(targetEntity=User::class, inversedBy="tasks")
 * @ORM\JoinColumn(nullable=true)
 *
 * @var User the task user
 */
private $user;
```

Création d'une relation « OneToMany » entre l'entité « Task » et l'entité « User »

```php
/**
 * @ORM\OneToMany(targetEntity=Task::class, mappedBy="user")
 *
 * @var Collection|Task[] the user tasks
 */
private $tasks;
```

Ajout de l'utilisateur actuellement connecté avant la persistance des données et la synchronisation en base de données.

```php
if ($form->isSubmitted() && $form->isValid()) {
    $user = $this->getUser();
    if (!$user instanceof User) {
        $user = $this->userService->userByDefault();
    }
    $task->setUser($user);

    $this->doctrine->getManager()->persist($task);
    $this->doctrine->getManager()->flush();

    $this->addFlash('success', 'La tâche a été bien été ajoutée.');

    return $this->redirectToRoute('task_list');
}
```

L'auteur d'une **tâche** n'est **pas modifiable**.

Afin de corriger les **tâches orphelines** (sans utilisateur attitré), j'ai créé une **commande** interne permettant de relier les tâches à un utilisateur nommé « **anonymous** ». Cette commande sera utile lors de la mise en production des modifications.

php bin/console tasks:linker

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console tasks:linker

Tasks linker
============


 [INFO] Found 4 tasks without user.

Details
-------

Task #54 linked to anonymous user.
Task #58 linked to anonymous user.
Task #59 linked to anonymous user.
Task #60 linked to anonymous user.


 [OK] Tasks linked to anonymous user.
```

➢ Utilisateurs

J'ai ajouté un champ « **rôles** » dans le **formulaire** de création et de modification d'un utilisateur. La sélection d'au moins un rôle est obligatoire.

Ajout du champ « roles » dans le formulaire « UserType.php »

```php
->add('roles', ChoiceType::class, [
    'choices' => [
        'Admin' => 'ROLE_ADMIN',
        'User' => 'ROLE_USER',
    ],
    'multiple' => true,
    'empty_data' => [],
    'expanded' => true,
    'label' => 'Rôles',
    'required' => true,
    'constraints' => [
        new \Symfony\Component\Validator\Constraints\NotBlank([
            'message' => 'Vous devez choisir au moins un rôle.',
        ]),
        ],
```

➢ Frontend

La navigabilité dans l'application étant compliqué, j'ai rajouté un **lien** vers la **page d'accueil** sur le nom de l'application en entête.

J'ai également rajouté un **lien** vers la liste des tâches sur le bouton « **consulter les tâches terminées** » car il ne pointait nulle part.

De plus, j'ai **mis à jour** la librairie **JQuery**. Cette mise à jour était nécessaire car certains scripts tiers peuvent présenter des failles de sécurité connues, faciles à identifier et à exploiter par des pirates informatiques.

J'ai également rajouté les **auteurs** dans la liste des **tâches**.

Enfin, j'ai **personnalisé** les **pages d'erreurs** afin qu'elles soient plus explicite pour les utilisateurs.

5) Implémentations des tests automatisés

Des **tests unitaires** et **fonctionnels** ont été créés pour l'application à l'aide de **PHPUnit**. Les tests couvrent plus de 98% de l'application et les différents scénarios sont traités.

D:\GitProjects\P8_ToDoAndCo\src / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	98.52%	200 / 203		96.55%	56 / 58		84.62%	11 / 13	
Command	100.00%	20 / 20		100.00%	3 / 3		100.00%	1 / 1	
Controller	98.48%	65 / 66		91.67%	11 / 12		75.00%	3 / 4	
DataFixtures	100.00%	30 / 30		100.00%	2 / 2		100.00%	1 / 1	
Entity	100.00%	39 / 39		100.00%	28 / 28		100.00%	2 / 2	
Form	100.00%	10 / 10		100.00%	2 / 2		100.00%	2 / 2	
Security	91.30%	21 / 23		87.50%	7 / 8		50.00%	1 / 2	
Service	100.00%	15 / 15		100.00%	3 / 3		100.00%	1 / 1	
Kernel.php	n/a	0 / 0		n/a	0 / 0		n/a	0 / 0	

Legend

Low: 0% to 35% **Medium**: 35% to 70% **High**: 70% to 100%

Generated by php-code-coverage 9.2.15 using PHP 7.4.27 and PHPUnit 9.5.21 at Mon Jul 11 8:20:06 UTC 2022.

Les dépendances **Faker** et **Liip** m'ont permis de **générer des données** factices et de **charger des fixtures** dans la base de données de test.

```
    "fakerphp/faker": "^1.19",
    "liip/test-fixtures-bundle": "^2.4",
```

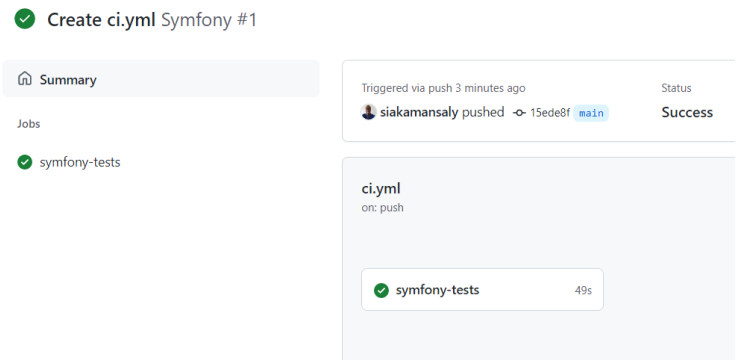
Les tests peuvent être lancés en local grâce à PHPUnit.

Commande PHPUnit : « php bin/phpunit »

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/phpunit
PHPUnit 9.5.21 #StandWithUkraine

Testing
.....................................                             38 / 38 (100%)

Time: 00:14.852, Memory: 78.00 MB

OK (38 tests, 92 assertions)

Generating code coverage report in HTML format ... done [00:00.220]
PS D:\GitProjects\P8_ToDoAndCo>
```

De plus, l'ensemble des tests ont été automatisés à l'aide de **Git Actions** et sont exécutés automatiquement après chaque modification de code.

Affichage tableau de bord Git Actions



Les **tests** sont executés dans **l'ordre suivant** (paramétrage du fichier PHPUnit) :

- Chargement des données de la base de données de test
- Exécution des tests unitaires
- Exécution des tests d'intégration
- Exécution des tests fonctionnels

Fichier de paramétrage phpunit.xml.dist

```
<testsuites>
    <testsuite name="Project Test Suite">
        <directory>tests/DataFixtures</directory>
        <directory>tests/Unit</directory>
        <directory>tests/Integration</directory>
        <directory>tests/Functional</directory>
    </testsuite>
</testsuites>
```

6) Analyse du code

L'ensemble du **code** a été **vérifié** à la suite des corrections et nouvelles implémentations.

➢ Dépendances

Le fichier des **dépendances** de l'application est maintenant valide. Pour corriger celui-ci, une description et un Namespace ont été intégrés.

```
PS D:\GitProjects\P8_ToDoAndCo> composer valid
./composer.json is valid
```

➢ Base de données

Le mappage des fichiers **doctrine** est toujours fonctionnel et la **base de données** se synchronise toujours bien avec les fichiers doctrine.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console doctrine:schema:validate

Mapping
-------

[OK] The mapping files are correct.

Database
--------

[OK] The database schema is in sync with the mapping files.
```

➢ Fichiers de code, views et de configuration

L'ensemble des fichiers ont été documentés grâce à la **PHPDoc**.

```
49      /**
50       * Task creation page.
51       *
52       * @Route("/tasks/create", name="task_create")
53       *
54       * @return Response|RedirectResponse
55       */
56      public function createAction(Request $request): Response
57  >   {…
80      }
```

L'analyse **PHPStan** de niveau 9 nous révèle maintenant aucunes erreurs. En effet, la mise à jour de la version PHP (version 7.4.0) du Framework nous a permis de pouvoir effectuer la déclaration de type au niveau des fonctions, valeurs de retour et propriétés de classes.

```
PS D:\GitProjects\P8_ToDoAndCo> vendor/bin/phpstan analyse
Note: Using configuration file D:\GitProjects\P8_ToDoAndCo\phpstan.neon.
 27/27 [============================] 100%

[OK] No errors
```

L'ensemble des fichiers **Twig** ont toujours une syntaxe valide.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console lint:twig templates

[OK] All 9 Twig files contain valid syntax.
```

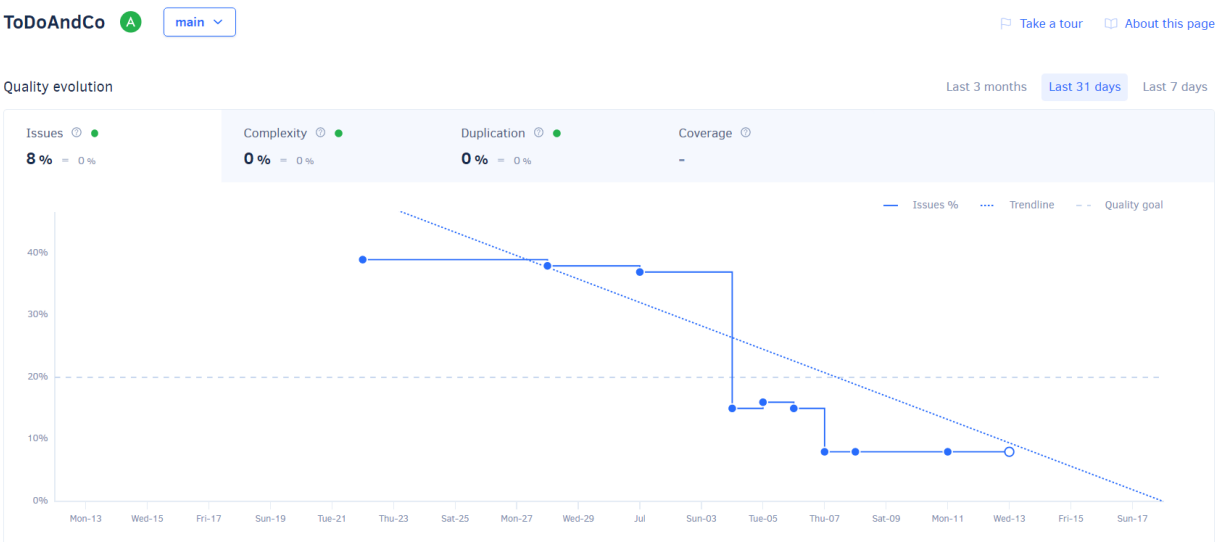
L'ensemble des fichiers **YAML** ont toujours une syntaxe valide.

```
PS D:\GitProjects\P8_ToDoAndCo> php bin/console lint:yaml config

[OK] All 22 YAML files contain valid syntax.
```

➢ Analyse complète

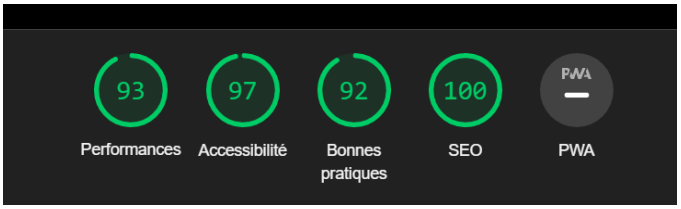
Le code de l'application ToDo & Co a été analysé entièrement sous **Codacy** et dispose désormais d'une excellente notation (**A**). Le **taux d'erreur** est passé de **39%** à **8%**.

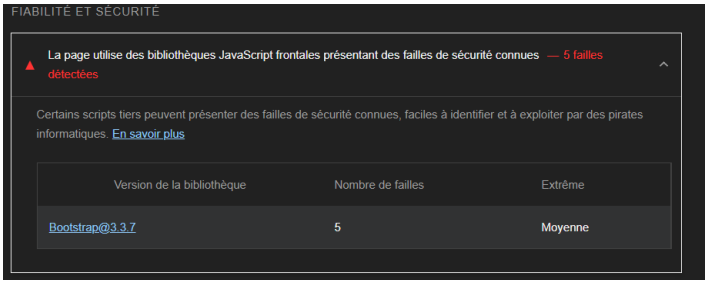
Lien : https://app.codacy.com/gh/siakamansaly/Audit-and-Improve-Symfony-App/dashboard



7) Analyse du Frontend

L'analyse grâce à l'outil Google **Lighthouse** est toujours bonne cependant elle nous rappelle qu'une mise à jour de Bootstrap permettrait d'éviter des failles de sécurité.





L'analyse **W3C** du code **HTML** ne dispose plus d'erreurs à la suite de la correction de l'attribut <nav>.

L'analyse **W3C** du code **CSS** est toujours valide. Il y a toujours 6 erreurs remontées avec notamment des propriétés CSS inexistantes ou erronées.

En effet, je n'ai effectué aucunes actions au niveau du CSS. Un travail sur le design étant nécessaire ainsi qu'une mise à jour de Bootstrap. L'intervention d'un développeur Frontend serait idéale.

8) Analyse des performances

➢ Récapitulatif des optimisations

J'ai appliqué les **optimisations** préconisées par **Blackfire** mais également les **recommandations** de **Symfony** pour gagner en performance.

Tout d'abord, le **chargement de Composer** a été **optimisé** en production et la mise en **cache** des **annotations** de doctrine a été activée.

Commande : composer dump-autoload --no-dev --classmap-authoritative

```
PS D:\GitProjects\P8_ToDoAndCo> composer dump-autoload --no-dev --classmap-authoritative
Generating optimized autoload files (authoritative)
Generated optimized autoload files (authoritative) containing 4625 classes
```

L'option permettant au **conteneur de services** d'être vidé dans un seul fichier a été activé.

fichier « services.yaml »

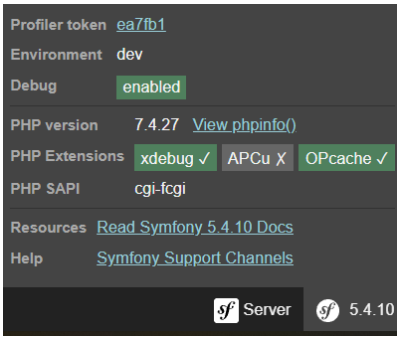
```
6   parameters:
7       container.dumper.inline_factories: true
```

J'ai également activé **OP Cache** pour maximiser les performances de l'application.

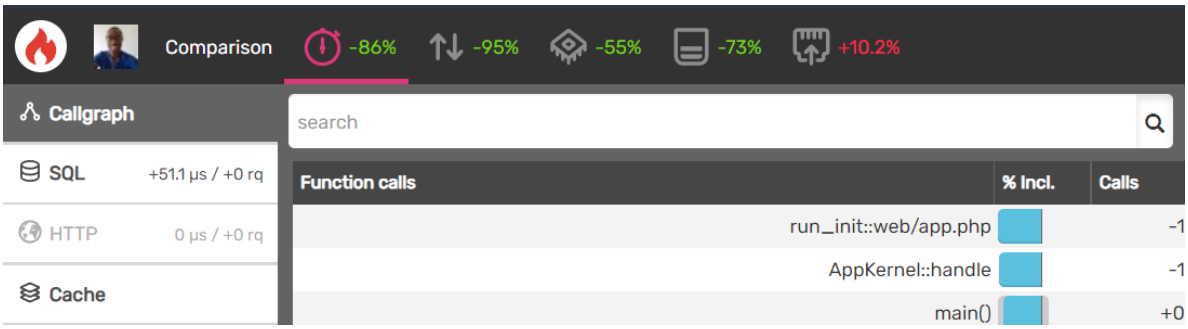
Fichier « php.ini »

```
[opcache]
zend_extension=opcache
opcache.enable=1
opcache.enable_cli=1
opcache.memory_consumption=256
opcache.max_accelerated_files=20000
opcache.validate_timestamps=0
```

OPcache activé



➢ Comparatif général des performances



Les différentes **routes** de l'application ont de nouveau été **profilé** avec Blackfire.

On constate un **gain** énorme en termes de **performance**. En effet, le temps d'accès à une page a été divisé par 7 en moyenne. Cela est également dû à la montée de version de Symfony et PHP.

Comparatif de temps entre l'application initiale et l'application modifiée

Temps global (ms)			
URL (Routes)	**Avant**	**Après**	**Différence**
/	254,00 ms	35,20 ms	-86%
/tasks	262,00 ms	40,00 ms	-85%
/tasks/create	315,00 ms	43,30 ms	-86%
/tasks/{id}/edit	320,00 ms	60,00 ms	-81%
/login	171,00 ms	24,80 ms	-85%
/users	271,00 ms	47,80 ms	-82%
/users/create	292,00 ms	49,10 ms	-83%
/users/{id}/edit	339,00 ms	52,10 ms	-85%

➢ Plus de détails

Exemple de rapport blackfire détaillé

Routes	Avant	Après
/tasks	https://blackfire.io/profiles/c15b9222-5df6-4fb8-b327-4b260f696210/graph	https://blackfire.io/profiles/944633fc-6c32-4ed4-92ab-8db2e839e448/graph
/login	https://blackfire.io/profiles/6fe0a35f-d258-478f-99ba-2033a43c3e2b/graph	https://blackfire.io/profiles/6deaca28-ee00-42a4-91bd-033796ee2cf0/graph
/	https://blackfire.io/profiles/f25924b4-dbf0-4079-833d-e18b8762d4d3/graph	https://blackfire.io/profiles/c988210b-4b55-4b9f-ac6a-4b38a816db3a/graph
/users	https://blackfire.io/profiles/e31db73d-f5e1-451c-9716-c5d093f5ea33/graph	https://blackfire.io/profiles/30f948d3-a004-4abe-a546-a34db41af7ad/graph

Temps d'entrée / sortie (ms)			
URL (Routes)	Avant	Après	Différence
/	203,00 ms	13,40 ms	-93%
/tasks	204,00 ms	9,52 ms	-95%
/tasks/create	245,00 ms	11,70 ms	-95%
/tasks/{id}/edit	246,00 ms	14,20 ms	-94%
/login	134,00 ms	5,77 ms	-96%
/users	214,00 ms	11,00 ms	-95%
/users/create	228,00 ms	12,10 ms	-95%
/users/{id}/edit	258,00 ms	11,30 ms	-96%

Temps du CPU (ms)			
URL (Routes)	Avant	Après	Différence
/	51,30 ms	21,80 ms	-58%
/tasks	58,00 ms	30,50 ms	-47%
/tasks/create	69,90 ms	31,60 ms	-55%
/tasks/{id}/edit	74,70 ms	45,80 ms	-39%
/login	37,10 ms	19,10 ms	-49%
/users	57,30 ms	36,80 ms	-36%
/users/create	64,30 ms	37,00 ms	-42%
/users/{id}/edit	81,70 ms	40,90 ms	-50%

Mémoire consommée (MB)			
URL (Routes)	Avant	Après	Différence
/	12,20 MB	3,28 MB	-73%
/tasks	12,30 MB	3,40 MB	-72%
/tasks/create	15,80 MB	4,24 MB	-73%
/tasks/{id}/edit	15,80 MB	4,44 MB	-72%
/login	7,94 MB	2,77 MB	-65%
/users	12,20 MB	3,35 MB	-73%
/users/create	15,70 MB	4,67 MB	-70%
/users/{id}/edit	15,80 MB	4,69 MB	-70%

E. CONCLUSIONS GENERALES

En conclusion, la dette technique de l'application a bien été réduite. L'analyse globale de Codacy nous le confirme grâce à une excellente note de A ainsi qu'un taux d'erreur qui a baissé de plus de 30%.

Les corrections et nouvelles implémentations demandées ont bien été réalisées et apporteront une plus-value à l'application.

De plus, l'intégration des tests automatisés permettront de continuer à suivre les bonnes pratiques, d'anticiper les régressions liées au changement, et de vérifier si tout fonctionne normalement.

L'application répond également aux attentes en termes de performance. Cependant, il est important de rester vigilant sur ce sujet. Il serait pertinent de mettre en place des tests de performance.

Les recommandations concernant le frontend n'ont pas été réalisées et nécessiteraient l'intervention d'un développeur Frontend. C'est pourquoi, en complément de cet audit, je préconise que cette application fasse l'objet d'un audit de qualité Web afin de voir s'il répond à des engagements en matière d'assurance qualité et de respect de l'utilisateur (référentiel Opquast).