Audit de qualité

ToDo & Co est une startup dont le cœur de métier est une application de gestion des tâches quotidiennes.

L'entreprise vient tout juste d'être crée, et l'application a dû être développée rapidement pour permettre de présenter à de potentiels investisseurs que le concept est viable (Minimum Viable Product ou MVP) avec le Framework Symfony dans sa version 3.1.

La mission qui m'a été confier et de développer de nouvelles fonctionnalités, et d'analyser la qualité du code ainsi que ses performances.

Ce document explique les améliorations apportées au code et compare les performances avant et après modifications.

Sommaire

1 – Le code initial

- *Premier audit du code initial*
  - *1.1 Rapport CodeClimate et Codacy*
  - *1.2 Rapport PHPMetrics et PHPCS*
  - *1.3 Version de Symfony et PHP*
  - *1.4 La sécurité*
  - *1.5 Mesure de performance BlackFire du code initiale*

2 – Améliorations

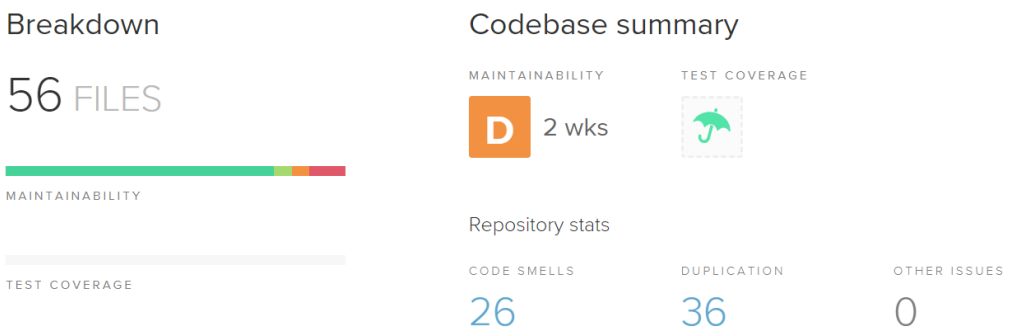
- *2.1 Coté Backend*
  - *Principe SOLID*
  - *Rapport PHPMetrics*
  - *Redis pour le cache*
  - *Optimisation des paramètres de PHP*
  - *Nouvelles mesures de performance BlackFire*
- *2.2 Coté Frontend*

3 – Tests unitaires et fonctionnels

- *PHP Unit*

- 1 – Le code initial

1.1 Rapport CodeClimate et Codacy :

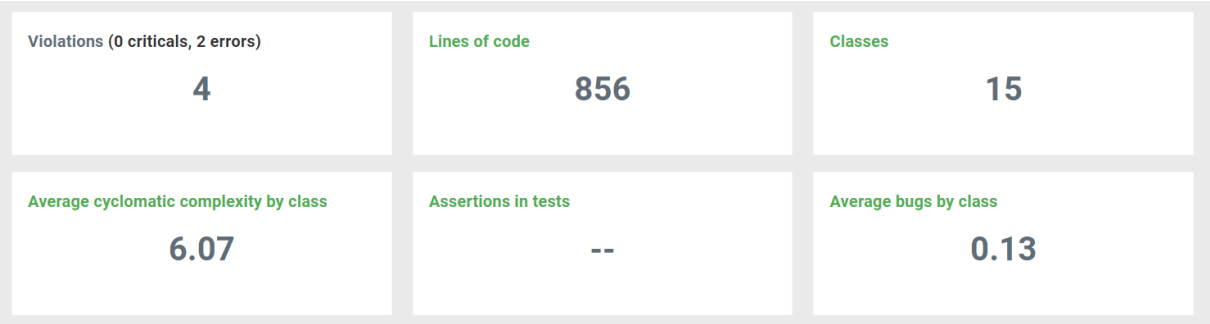


Des analyses de qualité de codes on été effectuer avec CodeClimate et Codacy.

Ces analyses n'on pas révéler de problèmes particulier sur le code de l'application. Il y'a toutefois quelques remonté sur le code du framework Symfony, elles ont été ignorées.

PSR1 et PSR2 sont respecter mais la PHPDoc est absente, elle très importante pour une meilleur compréhension et pour une meilleurs maintenance du code. De plus elle servira aux futurs développeurs qui pourraient reprendre l'application pour la faire évoluer.

1.2 Rapport PHPMetrics et PHPCS



Le rapport montre un taux de complexité des classes de 6.07 qui doit être abaissé au maximum.

PHPCS scan le code de chaque classe est donne un rapport très détaillé sur les éventuelles erreurs de code et corrections qui doivent être apportés.

Indentation, ouverture et fermeture de parenthèses, annotations peuvent être corrigé grâce à l'outil intégré PHPCBF. Certaines corrections doivent être faite à la main.

```
D:\TodoList (master -> origin)
λ phpcs d:/todolist/src/AppBundle/Controller/DefaultController.php

FILE: D:\TodoList\src\AppBundle\Controller\DefaultController.php
----------------------------------------------------------------------
FOUND 5 ERRORS AFFECTING 5 LINES
----------------------------------------------------------------------
```

```
D:\TodoList (master -> origin)
λ phpcs d:/todolist/src/AppBundle/Controller/SecurityController.php

FILE: D:\TodoList\src\AppBundle\Controller\SecurityController.php
----------------------------------------------------------------------
FOUND 12 ERRORS AFFECTING 11 LINES
----------------------------------------------------------------------
```

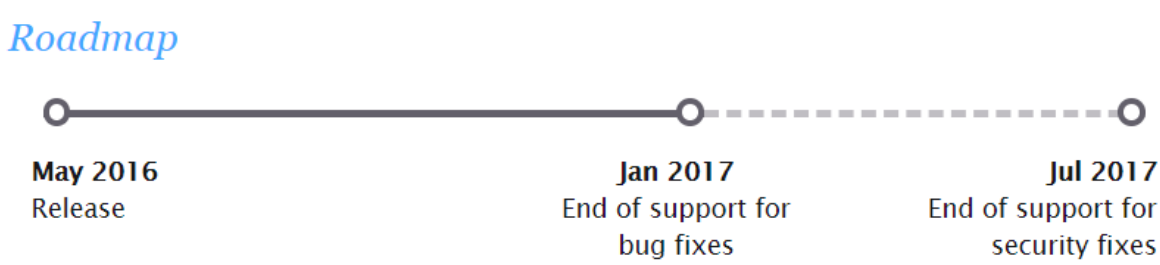
```
D:\TodoList (master -> origin)
λ phpcs d:/todolist/src/AppBundle/Controller/TaskController.php

FILE: D:\TodoList\src\AppBundle\Controller\TaskController.php
----------------------------------------------------------------------
FOUND 20 ERRORS AND 3 WARNINGS AFFECTING 18 LINES
----------------------------------------------------------------------
```

```
D:\TodoList (master -> origin)
λ phpcs d:/todolist/src/AppBundle/Controller/UserController.php

FILE: D:\TodoList\src\AppBundle\Controller\UserController.php
----------------------------------------------------------------------
FOUND 12 ERRORS AND 5 WARNINGS AFFECTING 14 LINES
----------------------------------------------------------------------
```

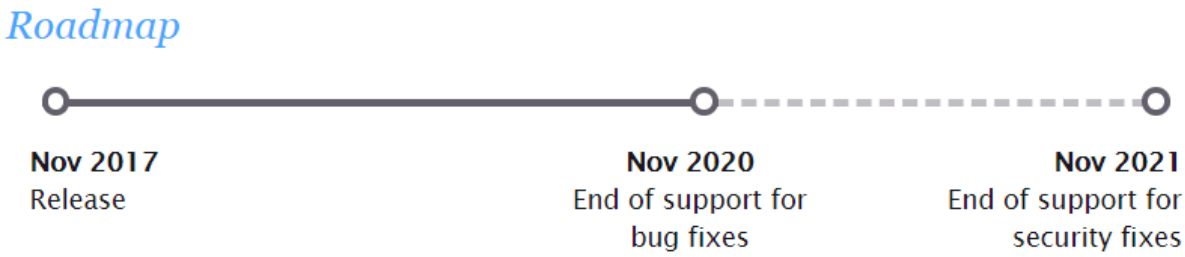
1.3 Version de Symfony et PHP

La version de Symfony initial de l'application est la 3.1.

Cette version à été mis en ligne en mai 2016 puis maintenu à jour jusqu'en janvier 2017 pour les corrections de bugs et jusqu'en juillet 2017 pour les correctifs de sécurité.



Passer sur la version 3.4 (*LTS = Long Term Support*) de Symfony permet de recevoir des mises à jour jusqu'en Novembre 2020 et des correctifs de sécurité jusqu'en Novembre 2021.



Seconde optimisation : Upgrader la version de PHP

La version de PHP utilisé est la 5.5.9, passer à la version 7.2 de PHP augmentera les performances au niveau de l'utilisation des ressources et offrira la possibilité d'utilisé les dernières fonctionnalités de PHP comme le typage des variables qui rendra le code plus stable.

Comparaison des performances de Symfony 3.1 avec PHP 5 et la version 3.4 de Symfony avec PHP 7.2 d'après le site php benchmarks : http://www.phpbenchmarks.com/fr/benchmark/apache-bench/php-7.2/select-version/symfony.html

1.4 La sécurité : Le niveau de sécurité initial est insuffisant, il permet d'accéder à certaines pages sans être authentifié.

La page qui liste les utilisateurs **/users** et qui permet leur modification **/users/{id}/edit** est accessible sans authentification. Un correctif doit être apporté pour que cette page soit accessible seulement à un utilisateur ayant les droits **ROLE_ADMIN**.

Des améliorations sont à apportés au formulaire d'authentification, il n'est pas protéger contre la faille CSRF.

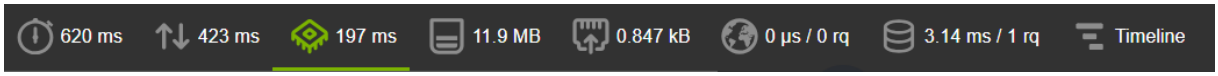
Réf : https://symfony.com/doc/current/security/form_login_setup.html

1.5 Mesure de performance BlackFire du code initiale

1 Mesure BlackFire de la page Home

	Avant modifications (windows)		Avant modifications (linux)	
Route	Temps ms	Mémoire mb	Temps ms	Mémoire mb
/tasks	224	9.43	55.7	3.53
/users	315	12.2	56.5	3.53
/users/create	288	12.9	67.5	4.03
/tasks/create	219	9.43	69.8	4.08
/login	249	9.71	48.5	3.21
/	241	9.42	49.8	3.01

Grace aux données récoltées par BlackFire on peut comprendre pourquoi une page mettra 224 ms pour s'afficher et consommera 9.43 mb de mémoire.

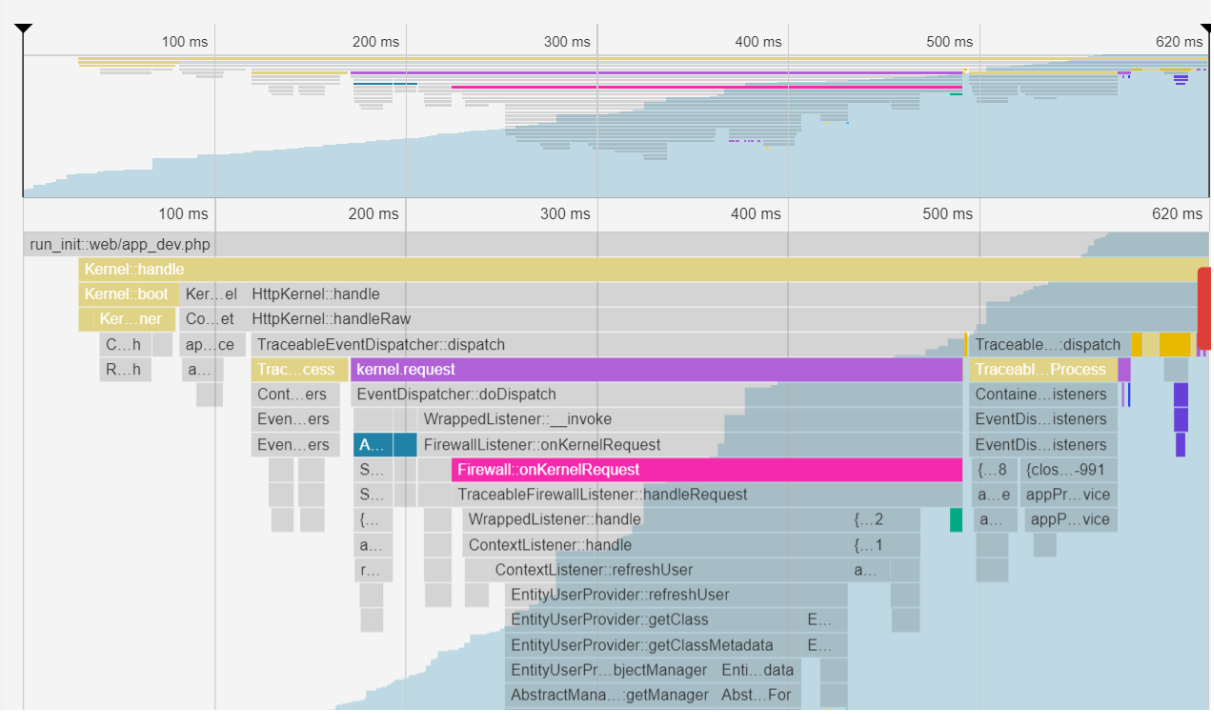


Temps d'accès, exécution processeur, occupation en mémoire, temps de la requête sql nous pouvons voir ce qu'il peut être amélioré pour rendre l'application plus réactive.

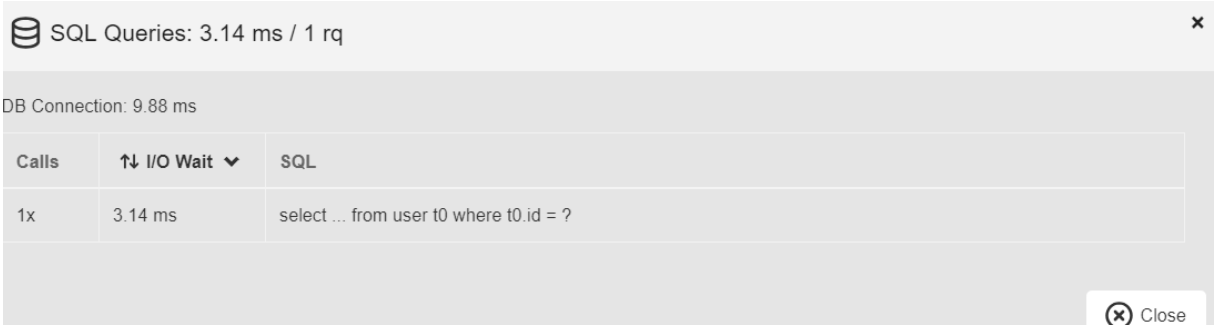
Une étude démontre que 40% des utilisateurs quitte un site si la page met plus de 3 secondes à s'afficher.

De ce fait il y'a peu de chance qu'un utilisateur revienne sur cette page.

D'où l'importance de prendre des mesures et d'améliorer certaines fonctions si il y'en a vraiment besoins.



La Timeline permet de voir le déroulement des actions que le système effectue pour afficher une page.



Nous pouvons voir le nombre de requête sql et leur temps d'exécution.

2 – Améliorations

2.1 Améliorations coté Backend

- Principe S.O.L.I.D.

Un des principes SOLID dit qu'une classe, une fonction ou une méthode doit avoir une et une seule responsabilité. Par conséquent es contrôleurs on était modifié.

Les classes TaskController et UserController s'occupe uniquement du CRUD, la logique métier des méthodes traitant les données de formulaires à été déplacer dans des **Handler** (AppBundle/FormHandler).

Les méthodes s'occupant d'afficher la liste des tâche ou des utilisateurs on dorénavant leur propre contrôleur.

- Les contrôleurs étendaient de la classe **Controller** (classe déprécié sur les nouvelles versions de Symfony elle et remplacer par la classe **AbstractControler)** .

La classe Controller à donc été supprimer, mais pour de meilleur performance je n'ai pas étendu la classe vers sa remplaçante mais plutôt choisi d'injecter les services vraiment utiles aux contrôleurs.

- Ajout de la PHPDOC pour une meilleure lisibilité du code.

> You can extend either `Controller` or `AbstractController`. The difference is that when you extend `AbstractController`, you can't access to your services via `$this->get()` or `$this->container->get()`, only to a set of common Symfony services. This forces you to write more robust code to access services.
>
> Moreover, in Symfony 4.2 `Controller` was deprecated in favor of `AbstractController`, so using the latter will make your apps future-proof.

> **New in version 3.3**: The `AbstractController` class was added in Symfony 3.3.

Réf : https://symfony.com/doc/3.4/controller.html#the-base-controller-classes-services

```php
public function createAction(Request $request)
{
    $task = new Task();
    $form = $this->createForm(TaskType::class, $task);

    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();

        $em->persist($task);
        $em->flush();

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

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

    return $this->render('task/create.html.twig', ['form' => $form->createView()]);
}
```

2 - Contrôleur avant modification

```php
public function createTask(Request $request, CreateTaskHandler $createTaskHandler)
{
    $task = new Task();

    $form = $this->formFactory->create( type: TaskType::class, $task)
        ->handleRequest($request);

    if ($createTaskHandler->handle($form, $task)) {

        return new RedirectResponse($this->urlGenerator->generate( name: 'task_list'),
            status: RedirectResponse::HTTP_FOUND);
    }

    return new Response($this->twig->render( name: 'task/create', [
        'form' => $form->createView()
    ]), status: Response::HTTP_OK);
}
```

```php
public function handle(FormInterface $form, Task $task)
{
    if ($form->isSubmitted() && $form->isValid()) {

        $task->setUser($this->tokenStorage->getToken()->getUser());

        $this->repository->save($task);

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

        return true;
    }
    return false;
}
```

3 - Contrôleur et son handler après modification

- Modification du **SecurityController** :
  Suppression des méthodes sans retour **Login_Check** et **Logout** qui indiqué seulement la route pour la vérification du login et celle de la déconnexion. Les routes on été ajouté dans le fichier **routing.yml** (app/config/routing.yml).

- **Modification du formulaire d'authentification Login Form:**

  Comme mentionné dans l'audit de code initiale le formulaire ne respect pas les bonnes pratiques de Symfony

  Modifications apportés :

  Vérification du CSRF Token au moment de l'authentification : permet de protéger les données envoyées par un utilisateur.

```html
<input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
```

Pour que ce processus fonctionne il a fallu également ajouter la classe **LoginFormAuthenticator** qui étend de la classe **AbstarctFormLoginAuthenticator** qui permet de vérifier toute les données reçu comme le nom d'utilisateur, le mot de passe crypté …

- Correction de l'anomalie à l'ajout d'une tâche :
  Ajout d'une relation entre l'entité **Task** et **User**

```php
/**
 * @ORM\ManyToOne(targetEntity="AppBundle\Entity\User", inversedBy="task")
 * @ORM\JoinColumn(onDelete="cascade")
 */
private $user;
```

```php
/**
 * @ORM\OneToMany(targetEntity="AppBundle\Entity\User", mappedBy="user", orphanRemoval=false)
 */
private $task;
```

Dorénavant lors de l'ajout d'une tâche celle-ci est rattaché à l'utilisateur connecté (donc son auteur).

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

    $task->setUser($this->tokenStorage->getToken()->getUser());

    $this->repository->save($task);
```

1 On récupère l'utilisateur actuellement connecté pour l'associé à la tâche avant sa sauvegarde en BDD

- Avec cette relation entre l'utilisateur et ses tâches celui-ci peut *visualiser, modifier, supprimer et marquer* **uniquement** ses tâches. Une vérification est faite avant chaque action, le système vérifie que la tâche appartient bien à l'utilisateur connecté.

```php
if ($this->authorization->isGranted( attributes: TaskVoter::EDIT, $task) === true) {
```

2 Exemple avec la fonction de suppréssion d'une tâche . On vérifie que l'utilisateur connecté est bien l'auteur de la tâche

- Type Hinting :
  La version initial de PHP (5) ne permettait pas de typé les retours de méthodes ou des paramètres. Le typage à été ajouté suite au passage de la version 7 de PHP.

- Validation password:
  Ajout de la validation sur le mot de passe utilisateur car aucune été paramétré initialement.

- La commande **CreateAdminCommand** (dossier : AppBundle/Command/) pour facilité l'ajout d'un administrateur à l'application en ligne de commande.

```
D:\projet8 (master -> origin)
λ php bin/console app:create-admin administrateur password administrateur@email.com


You are about to create an admin-user.
Username: administrateur
Password: password
Email: administrateur@email.com
Role: ROLE_ADMIN
Admin successfully created
```

- Optimisation des paramètres de PHP

  Les recommandations de Symfony sont de modifié quelques paramètres du fichier PHP.ini pour optimiser le chargement.
  Réglage du RealPath et de l'OP Cache

```
; php.ini
; maximum memory allocated to store the results
realpath_cache_size=4096K

; save the results for 10 minutes (600 seconds)
realpath_cache_ttl=600
; maximum memory that OPcache can use to store compiled PHP files
opcache.memory_consumption=256

; maximum number of files that can be stored in the cache
opcache.max_accelerated_files=20000
```

  Réf : https://symfony.com/doc/3.4/performance.html

- Mise en place du cache avec **REDIS**

  Pour des performances accrues la mise en cache des données est primordiale.

  Le cache **REDIS** à été mis en place pour optimiser les requêtes Doctrine et obtenir un affichage rapide de la liste des tâches.

  Plusieurs caches sont gérés par Redis, le metadataCache, le queryCache et le resultCache.

  Paramétrage du metadataCache et queryCache : fichier App/Config/config.yml

```
orm:
    auto_generate_proxy_classes: "%kernel.debug%"
    naming_strategy: doctrine.orm.naming_strategy.underscore
    auto_mapping: true
    metadata_cache_driver: redis
    query_cache_driver: redis
```

Création du fichier : App/Config/redis.yml ou l'on indique l'adresse du serveur Redis.

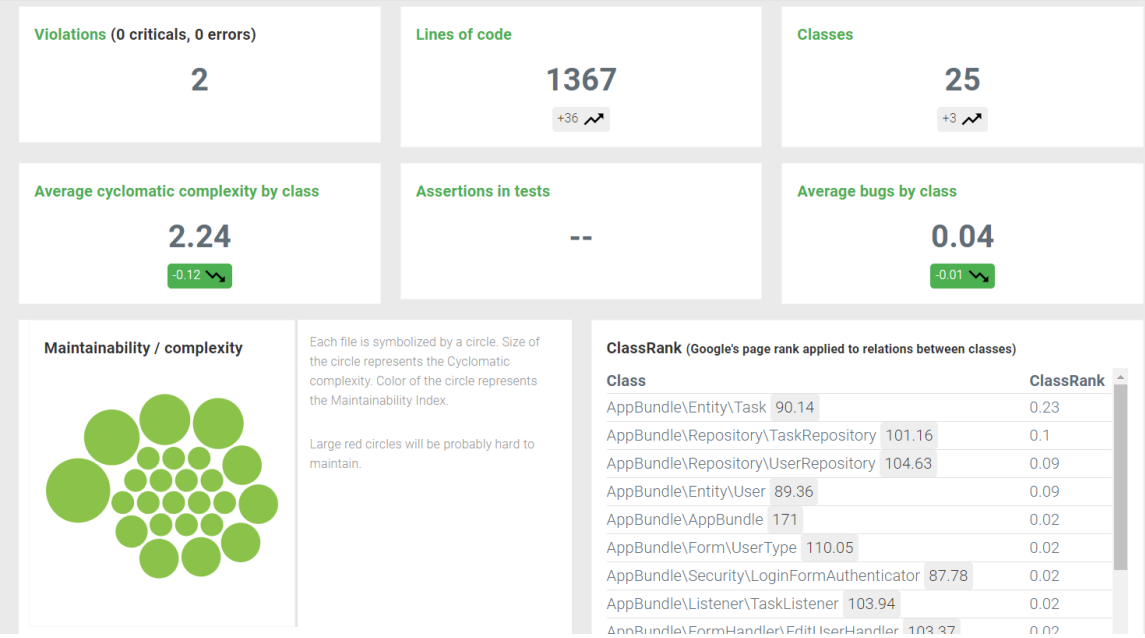
```
snc_redis:
    clients:
      default:
        type: predis
        alias: default
        dsn: redis://127.0.0.1:6379
      doctrine:
        type: predis
        alias: doctrine
        dsn: redis://127.0.0.1:6379
    doctrine:
      metadata_cache:
        client: doctrine
        entity_manager: default
        document_manager: default
      result_cache:
        client: doctrine
        entity_manager: default
      query_cache:
        client: doctrine
        entity_manager: default
```

Pour la mise en cache des données récupérer avec Doctrine il l'indiqué dans une requête (DQL ou avec le QueryBuilder). AppBundle/Repository/TaskRepository

```
public function taskList($user)
{
    return $this->createQueryBuilder( alias: 't')
        ->orderBy( sort: 't.createdAt', order: 'DESC')
        ->where( predicates: 't.user=:user')
        ->setParameter( key: 'user', $user)
        ->getQuery()
        ->useResultCache( bool: true)
        ->setQueryCacheLifetime( timeToLive: 60)
        ->setResultCacheId( id: 'tasks')
        ->getResult();
}
```

Nous demandons la liste des tâches, le résultat est mis en cache pendant 1 minute

Rapport PHPMetrics



Le nouveau rapport de PHPMetrics montre dorénavant une très bonne maintenabilité des classes (représenté par les cercles vert), un résultat de complexité des classes passant de 6.07 à 2.24 grâce au respect du principe SOLID : une classe une responsabilité. *Deux erreurs apparaissent elles demandent de passé les classes entity et repository en classe abstraite (ce qui impossible pour le bon fonctionnement de l'application, je pense donc à un bug de PHPMetrics)*

Nouvelles mesures de performance BlackFire

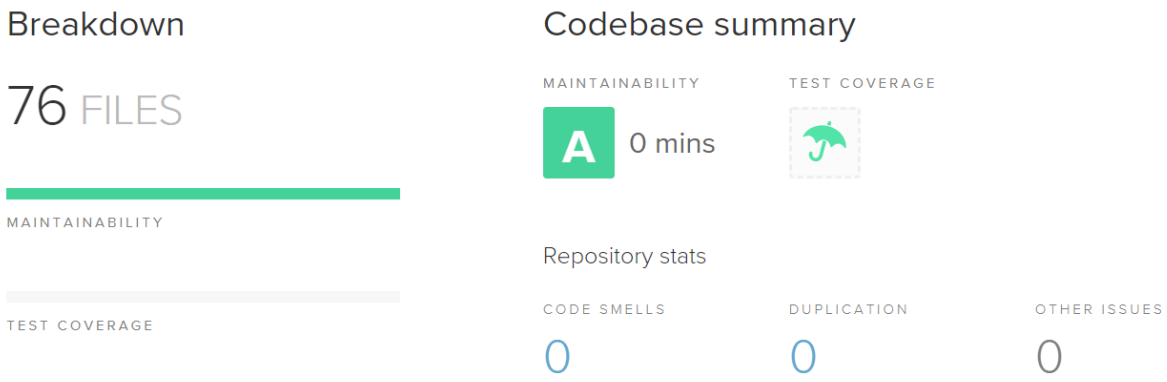
```
D:\projet8 (master -> origin)
λ blackfire curl http://127.0.0.1:8000
Profiling: [########################################] 10/10
Blackfire cURL completed
Graph URL https://blackfire.io/profiles/be793984-5939-45df-9706-c
No tests! Create some now https://blackfire.io/docs/cookbooks/tes
3 recommendations https://blackfire.io/profiles/be793984-5939-45d

Wall Time      222ms
I/O Wait       152ms
CPU Time      69.7ms
Memory       7.97MB
Network         n/a      n/a      n/a
SQL             n/a      n/a
```

4 Mesure BlackFire de page Home optimisé

	Après modifications (windows)		Après modifications (linux)	
Route	Temps ms	Mémoire mb	Temps ms	Mémoire mb
/tasks	208	7.96	37.8	3.12
/users	225	7.97	32.4	2.96
/users/create	341	14.3	41.4	3.86
/tasks/create	209	7.97	40.5	3.86
/login	275	9.65	29.4	2.90
/	222	7.97	28	2.87

Les nouvelles mesures montrent un léger gain au niveau du temps d'accès des différentes routes. Le passage à PHP 7 et Symfony 3.4 à eu son effet. L'application gagne en fluidité, la navigation est plus agréable surtout lors de l'affichage des tâches grâce à l'implémentation du cache REDIS, et des optimisations du fichier PHP.INI préconisé par Symfony.

- Rapport CodeClimate après les modifications



Avec les différentes modifications apportées au projet le nombre de fichiers passe de 56 à 76, ajout de nouveaux contrôleurs, listerner, handler, repository… et un code maintenable. Le code venant des différentes librairies y compris celle de Symfony à été ignorer des mesures.

2.2 Améliorations coté Frontend

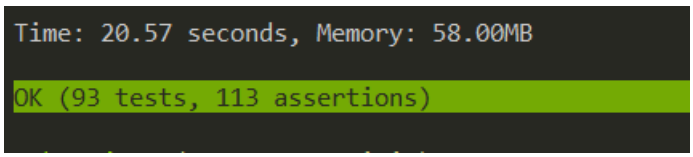
- La version de Bootstrap initial été la 3, je l'ai remplacé par la version 4 via CDN qui améliore le chargement et qui apporte les dernières nouveautés du langage CSS offrant un design plus moderne avec des fonctionnalités plus poussé.
- Suppression du code inutile dans le fichier Shop-homepage.css.
- Lorsqu'un administrateur est connecté deux liens apparaissent dans la barre de navigation : gérer les utilisateurs.
- Réorganisation de la page d'accueil. Alignement du titre, de l'image et des boutons.
- Réorganisation de l'affichage d'une tâche. Les boutons sont maintenant dans le cadre.
- Ajout d'une bannière obligatoire pour mentionner aux visiteurs que l'application utilise des cookies.
- Amélioration de l'ergonomie avec des titres de page dynamique.
- Ajout de la page tâche terminé :
  Le bouton « consulter les tâches terminées » été bien présent mais la page n'existé pas. Ajout d'une information : la date à laquelle la tâche à été marquer comme terminée.
- Ajout d'une page d'erreur.
- Réorganisation général de l'application pour la rendre responsive.

3 – Tests unitaires et fonctionnels

PHP Unit

La couverture des tests unitaires et fonctionnels avec PHPUnit couvrent 100% du code.
Différents test on été effectuer :
Réponses et redirection de chaque méthode des contrôleurs.
Scénario d'ajout, de modification et suppression d'une tâche.

Scénario d'ajout, de modification et suppression d'un utilisateur.



	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	314 / 314		100.00%	75 / 75		100.00%	23 / 23
Command		100.00%	29 / 29		100.00%	3 / 3		100.00%	1 / 1
Controller		100.00%	133 / 133		100.00%	15 / 15		100.00%	8 / 8
Entity		100.00%	45 / 45		100.00%	30 / 30		100.00%	2 / 2
Fixtures		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
FormHandler		100.00%	48 / 48		100.00%	10 / 10		100.00%	5 / 5
Form		100.00%	24 / 24		100.00%	4 / 4		100.00%	4 / 4
Listener		100.00%	8 / 8		100.00%	4 / 4		100.00%	1 / 1
Repository		100.00%	27 / 27		100.00%	9 / 9		100.00%	2 / 2
Security		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
AppBundle.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0