

AUDIT QUALITÉ DE CODE



Sommaire :

1. Rappel du contexte
2. Etat actuel de l'application
 - a. Framework utilisé
 - b. Problématiques
 - c. Qualité de code
 - d. Tests
3. Solutions apportées
 - a. Mise à jour de Symfony
 - b. Correction des anomalies et implémentation des fonctionnalités
 - c. Tests unitaires
 - d. Tests fonctionnels
 - e. Qualité de code
 - f. Outils de mesure de performances

1. Rappel du contexte

Todolist est une application web de gestion et planification de tâches. Cependant l'entreprise vient tout juste d'être montée, et l'application a dû être développée à toute vitesse pour permettre de montrer à de potentiels investisseurs que le concept est viable (on parle de Minimum Viable Product ou MVP). Suite à une levée de fond, l'application va pouvoir être développée. Il est donc demandé au développeur en charge du projet d'améliorer la qualité de l'application et de fournir un audit de qualité et de performance de l'application.

2. Etat actuel de l'application

a. Framework utilisé

L'application utilise la version 3 du framework. Cette version est sortie en novembre 2015 et n'est plus maintenue depuis juillet 2016. Il est donc impératif de migrer vers une version plus récente de Symfony.

b. Problématiques

Plusieurs anomalies ont été rapportées suite à au développement du MVP de l'application, voici la liste des ces dernières :

- Une tâche doit être attachée à un utilisateur :

Actuellement, lorsqu'une tâche est créée, elle n'est pas rattachée à un utilisateur. Il vous est demandé d'apporter les corrections nécessaires afin qu'automatiquement, à la sauvegarde de la tâche, l'utilisateur authentifié soit rattaché à la tâche nouvellement créée.

Lors de la modification de la tâche, l'auteur ne peut pas être modifié.

Pour les tâches déjà créées, il faut qu'elles soient rattachées à un utilisateur "anonyme".

- Choisir un rôle pour un utilisateur :

Lors de la création d'un utilisateur, il doit être possible de choisir un rôle pour celui-ci. Les rôles listés sont les suivants :

rôle utilisateur (ROLE_USER), rôle administrateur (ROLE_ADMIN).



Lors de la modification d'un utilisateur, il est également possible de changer le rôle d'un utilisateur.

c. Qualité de code

Le projet actuel répond en partie aux standards PSR, les classes sont structurées et il n'y a pas d'incohérences dans le code.

Pour ce qui concerne la sécurité, le composant Security utilisé pour la gestion de l'authentification est totalement obsolète et il est impératif de le mettre à jour.

L'analyse Codacy donne une note générale de B, qui est correcte mais avec 39% d'issues (le quality goal est de 20%) et 3% de complexité.

Repository name	Grade	Issues	Complexity
 todolist-start	 B	39%	3%

d. Tests

Aucun test unitaire ou fonctionnel n'est actuellement implémenté sur ce projet.

3. Solutions apportées

a. Mise à jour de Symfony

Pour garantir une bonne stabilité à l'application, il est préférable d'utiliser une version LTS de Symfony.

L'ancienne version utilisée n'était plus maintenue depuis juillet 2016.

Il a donc été décidé de migrer l'application vers la version 5.4, qui est récente et assez éprouvée pour garantir une bonne stabilité.


Les dépendances ont également été mises à jour.






















b. Correction des anomalies et implémentation des fonctionnalités

- Chaque tâche est rattachée à un utilisateur
- L'utilisateur ne voit désormais que ses tâches personnelles
- Les tâches ne peuvent être supprimées que par les auteurs de ces dernières
- Seuls les administrateurs peuvent supprimer les tâches anonymes
- Lors de la création de compte, un rôle est attribué à l'utilisateur
- La page de gestion des utilisateurs est accessible uniquement
- Les boutons "Se connecter" et "S'inscrire" sont adaptés en fonction de la page où se situe l'utilisateur

c. Tests unitaires

Les tests unitaires ont été réalisés sur les entités Task et User afin de tester tous les getters et les setters

















	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		100.00%	1 / 1		100.00%	13 / 13	CRAP		100.00%	21 / 21
Task		100.00%	1 / 1		100.00%	13 / 13	15		100.00%	21 / 21
__construct					100.00%	1 / 1	1		100.00%	3 / 3
getId					100.00%	1 / 1	1		100.00%	1 / 1
getCreatedAt					100.00%	1 / 1	1		100.00%	1 / 1
setCreatedAt					100.00%	1 / 1	1		100.00%	2 / 2
getTitle					100.00%	1 / 1	1		100.00%	1 / 1
setTitle					100.00%	1 / 1	1		100.00%	2 / 2
getContent					100.00%	1 / 1	1		100.00%	1 / 1
setContent					100.00%	1 / 1	1		100.00%	2 / 2
setIsDone					100.00%	1 / 1	1		100.00%	2 / 2
getIsDone					100.00%	1 / 1	1		100.00%	1 / 1
getUser					100.00%	1 / 1	2		100.00%	1 / 1
setUser					100.00%	1 / 1	2		100.00%	2 / 2
toggle					100.00%	1 / 1	1		100.00%	2 / 2

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		100.00%	1 / 1		100.00%	12 / 12	CRAP		100.00%	17 / 17
User		100.00%	1 / 1		100.00%	12 / 12	12		100.00%	17 / 17
getId					100.00%	1 / 1	1		100.00%	1 / 1
getUsername					100.00%	1 / 1	1		100.00%	1 / 1
setUsername					100.00%	1 / 1	1		100.00%	2 / 2
getPassword					100.00%	1 / 1	1		100.00%	1 / 1
setPassword					100.00%	1 / 1	1		100.00%	2 / 2
getEmail					100.00%	1 / 1	1		100.00%	1 / 1
setEmail					100.00%	1 / 1	1		100.00%	2 / 2
getRoles					100.00%	1 / 1	1		100.00%	2 / 2
setRoles					100.00%	1 / 1	1		100.00%	2 / 2
getSalt					100.00%	1 / 1	1		100.00%	1 / 1
eraseCredentials					100.00%	1 / 1	1		100.00%	1 / 1
getUserIdentifier					100.00%	1 / 1	1		100.00%	1 / 1

d. Les tests fonctionnels

Les tests fonctionnels sont définis comme une méthode permettant de tester la fonctionnalité d'une application logicielle. Le plus souvent, les tests fonctionnels sont utilisés pour vérifier des scénarios ou des modèles d'utilisation de bout en bout.

Resultats de test sur le TaskController :

	Classes and Traits			Functions and Methods				Lines		
Total		0.00%	0 / 1		0.00%	0 / 5	CRAP		64.81%	35 / 54
TaskController		0.00%	0 / 1		0.00%	0 / 5	55.45		64.81%	35 / 54
listAction					0.00%	0 / 1	2.06		75.00%	3 / 4
createAction					0.00%	0 / 1	4.01		92.86%	13 / 14
editAction					0.00%	0 / 1	8.51		80.00%	12 / 15
toggleTaskAction					0.00%	0 / 1	6.97		70.00%	7 / 10
deleteTaskAction					0.00%	0 / 1	42		0.00%	0 / 11

Et voici comment ces tests sont simulés :

```
public function testListAction()
{
    $client = static::createClient();
    $userRepository = static::getContainer()->get(UserRepository::class);

    // retrieve the test user
    $testUser = $userRepository->findOneByEmail('ruiz.nico64@gmail.com');

    // simulate $testUser being logged in
    $client->loginUser($testUser);

    // test e.g. the profile page
    $client->request('GET', '/tasks');
    $this->assertResponseIsSuccessful();
    $this->assertSelectorTextContains('h1', 'Liste de vos tâches');
}
```

```
public function testCreateAction()
{
    $client = static::createClient();
    $userRepository = static::getContainer()->get(UserRepository::class);
    $taskRepository = static::getContainer()->get(TaskRepository::class);

    // retrieve the test user
    $testUser = $userRepository->findOneByEmail('ruiz.nico64@gmail.com');

    // simulate $testUser being logged in
    $client->loginUser($testUser);

    $crawler = $client->request('GET', '/tasks/create');

    // select the button
    $buttonCrawlerNode = $crawler->selectButton('Ajouter');

    // retrieve the Form object for the form belonging to this button
    $form = $buttonCrawlerNode->form();

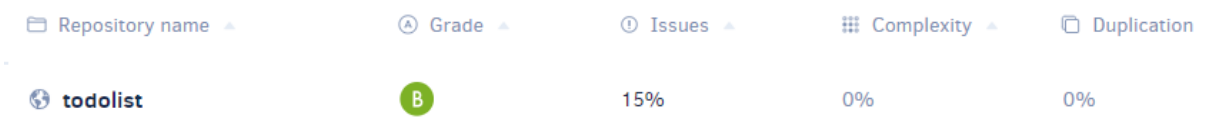
    // set values on a form object
    $form['task[title]'] = 'Titre test';
    $form['task[content]'] = 'Contenu test';

    // retrieve last created task
    $last = $taskRepository->findOneBy([], ['id' => 'desc']);

    // submit the Form object
    $client->submit($form);
    $this->assertSame('Titre test', $last->getTitle());
}
```

Ces tests visent principalement à simuler le parcours utilisateur sur le site et s'assurer que toutes les fonctionnalités de l'application fonctionnent comme elles sont supposées le faire.

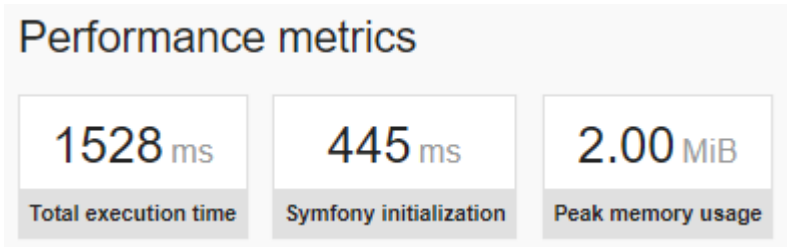
e. Qualité de code



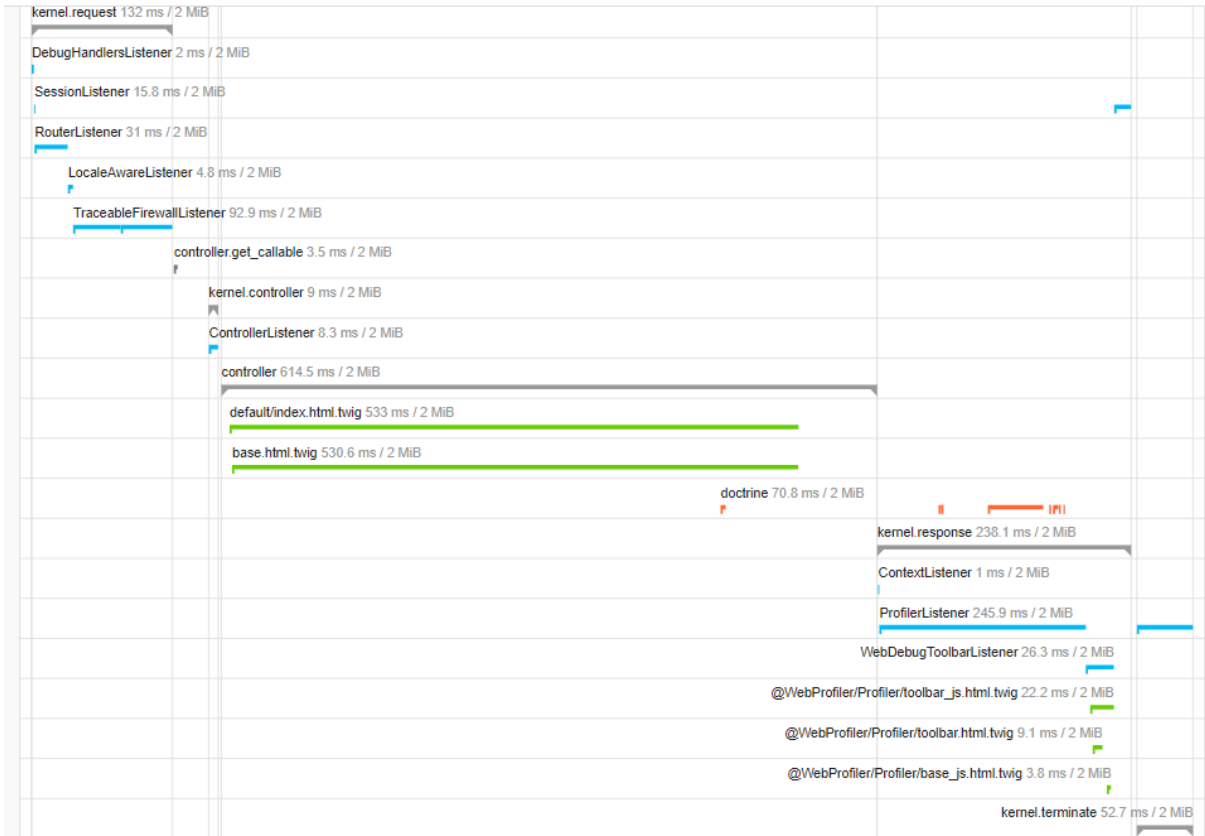
Une fois le projet mis à jour, l'analyse Codacy nous donne un score un score de B et 15% d'issues (en dessous du quality goal de 20%) et 0% de complexité.

f. Outils de mesures des performances

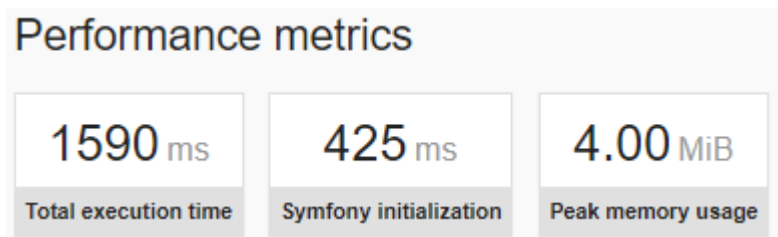
Page d'accueil :



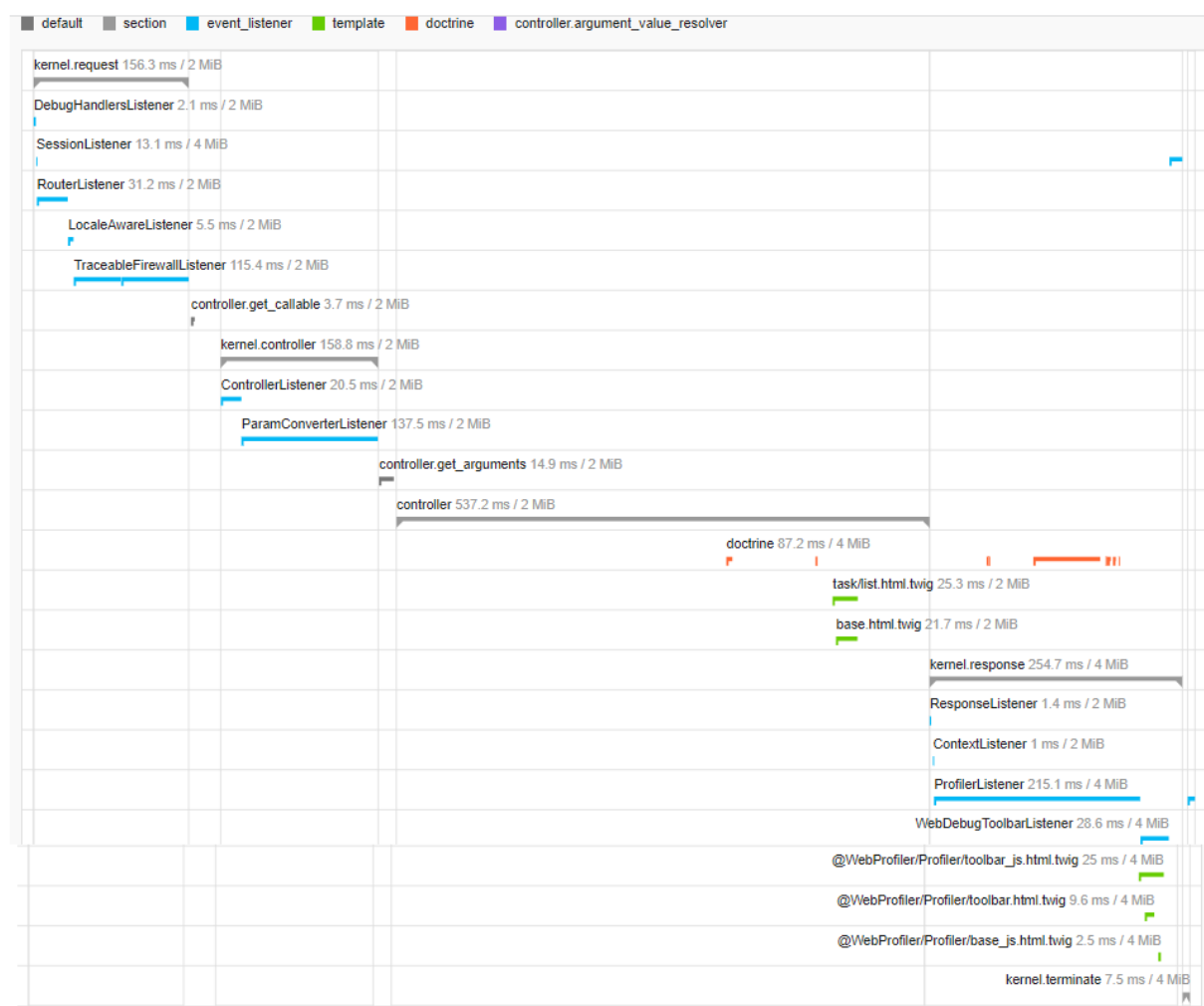
Chronologie d'exécution :



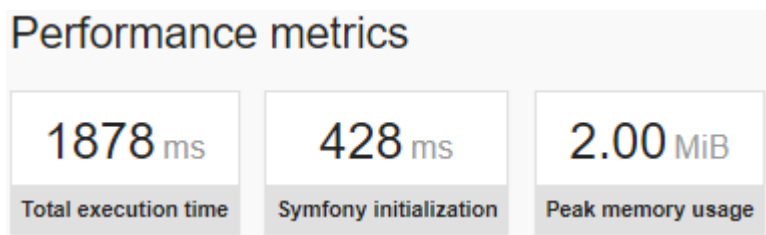
Page de liste des tâches :



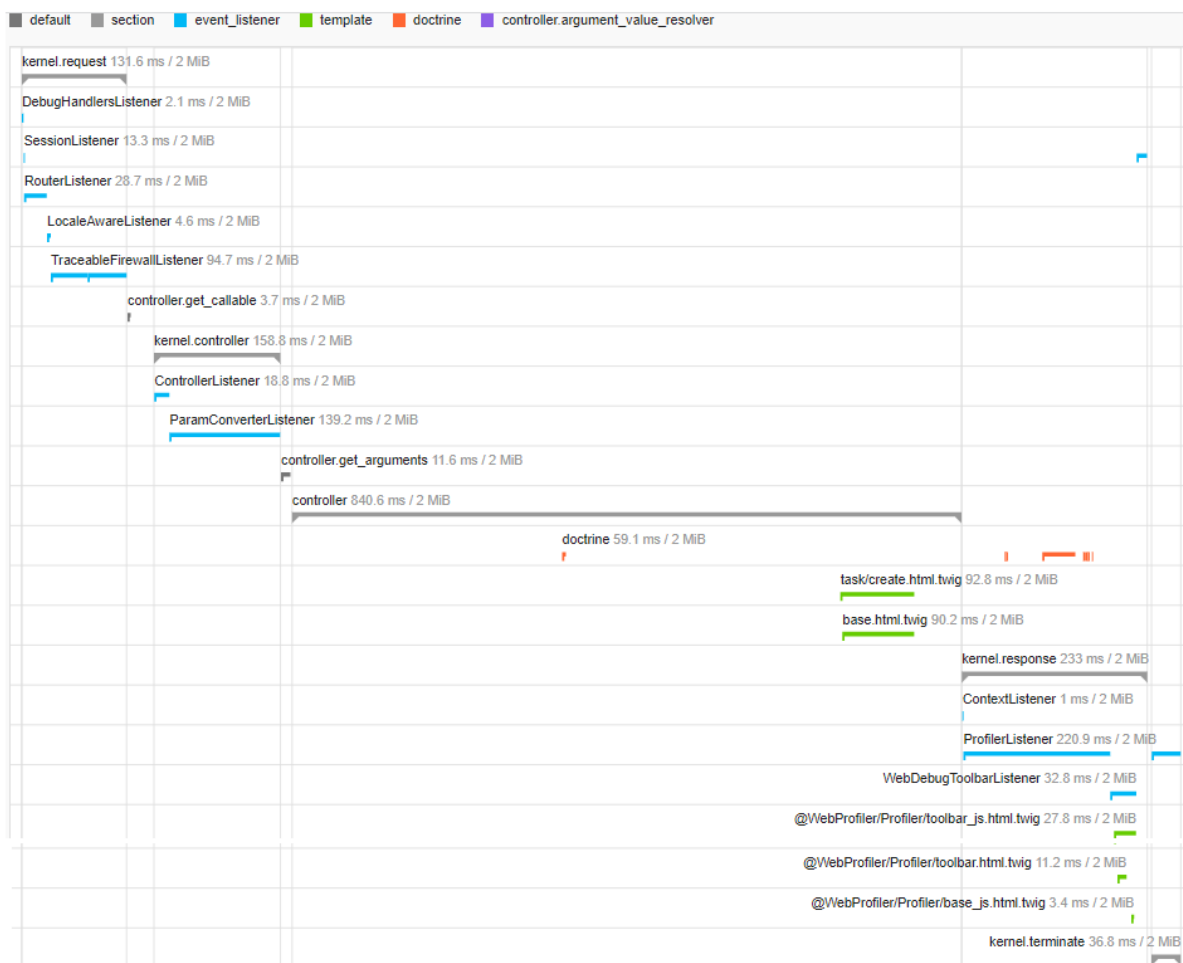
Chronologie d'exécution :



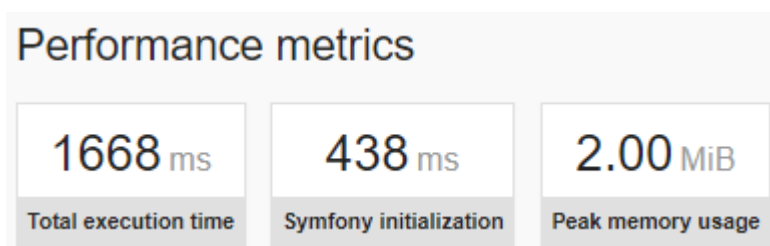
Page de création d'une tâche :



Chronologie d'exécution :



Page d'inscription :



Chronologie d'exécution :

