

Audit de qualité et de performance du projet toDoList

I. Etat des lieux technique de l'application initiale

A. Architecture du projet

L'application a été développée avec le Framework Symfony sous la version 3.1. Cette version n'est plus maintenue depuis juillet 2017.

Après analyse, il apparaît aussi que de nombreuses bibliothèques tierces qui ont été utilisées avec ce Framework sont actuellement dépréciées.

B. Dettes techniques et fonctionnelles

1. Analyse du client

En amont de la mise à jour de l'application le client dans son cahier des charges a déjà relevé plusieurs anomalies fonctionnelles ainsi que des améliorations à implémenter.

a) Anomalies

- ✓ 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.

b) Amélioration des « autorisations »

- ✓ Seuls les utilisateurs ayant le rôle administrateur (*ROLE_ADMIN*) doivent pouvoir 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 l'utilisateur "anonyme" peuvent être supprimées uniquement par les utilisateurs ayant le rôle administrateur (*ROLE_ADMIN*).

2. Mon analyse complémentaire

a) Fonctionnalités

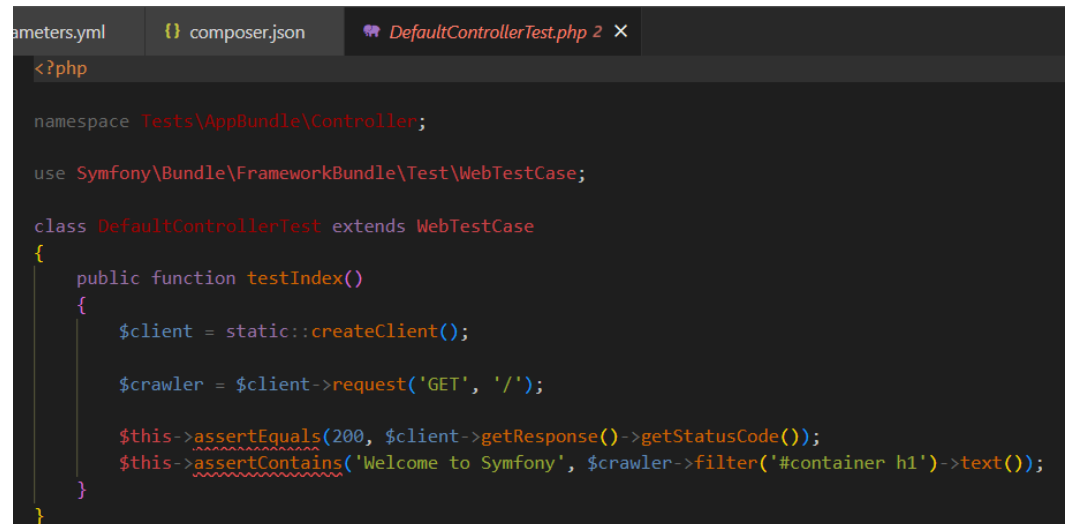
Suite à une analyse fonctionnelle de l'application complémentaire à celle du client, j'ai pu identifier les points suivants :

- ✓ Actuellement il n'y a aucun lien vers la page d'accueil.
- ✓ Sur la page d'accueil, le bouton « consulter la liste des tâches à faire » permet d'accéder non pas uniquement à la page des tâches à réaliser (comme on peut le penser) mais à toutes les tâches y compris celles qui ont déjà été validé. Cela ne semble pas être très pertinent.
- ✓ Absence de page pour les tâches marquées comme « Terminées » alors qu'il existe un lien en page d'accueil.
- ✓ Problème de traduction des formulaires comme celui pour la création de tâche.
- ✓ Absence de traduction des messages de validation et des erreurs.
- ✓ Absence de message flash sur certaines actions.

✓ Absence de pages d'erreurs en production.

b) Tests unitaires et fonctionnels

En observant le code initial on s'aperçoit qu'il n'y a qu'un seul test fonctionnel :



```
<?php
namespace Tests\AppBundle\Controller;
use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class DefaultControllerTest extends WebTestCase
{
    public function testIndex()
    {
        $client = static::createClient();

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

        $this->assertEquals(200, $client->getResponse()->getStatusCode());
        $this->assertContains('Welcome to Symfony', $crawler->filter('#container h1')->text());
    }
}
```

Il se contente de vérifier qu'en demandant la page d'accueil on obtienne bien un « h1 » avec comme texte « Welcome to Symfony » soit sans doute la page d'accueil avant la production de la moindre ligne de code.

Il n'y a bien évidemment aucun rapport de couverture de code.

II. Travaux effectuées sur l'application.

A. Architecture du projet initial

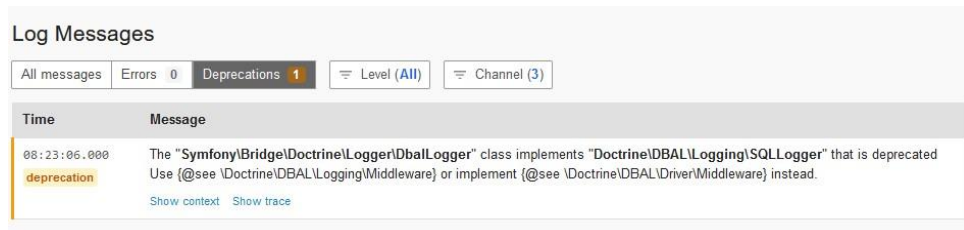
La version du Framework Symfony a été migrée vers la version 3.4 qui est la version stable la plus proche de l'initiale afin de réaliser l'audit de performance.

De plus toutes les bibliothèques tierces qui étaient dépréciées ont été mises à jour.

B. Architecture du projet après améliorations

J'ai opté pour la version Long Time Support (5.4) qui est complètement maintenue jusqu'en fin 2024 et 2026 pour la sécurité.

Il est à noter cependant que la bibliothèque suivante, n'a pas pu être mise à jour car ne dépendant pas des utilisateurs :



En effet cette bibliothèque est utilisée en interne au Framework Symfony et il faudra donc attendre que les concepteurs mettent à jour cette version pour que nous puissions à notre tour le faire également via la commande « *composer update* ».

L'ensemble des besoins (anomalies et améliorations) exprimés dans le cahier des charges par le client ont été réalisés.

A noté qu'un choix arbitraire a été fait de rediriger vers la page d'accueil (avec un message flash) un utilisateur qui souhaiterait accéder à une ressource qui lui est interdite (ex : un utilisateur qui souhaiterait accéder à la page « /users »). Dans le fichier « security.yaml » a été en effet intégré au niveau du « firewalls » l'instruction suivante :

```
access_denied_handler: App\Security\AccessDeniedHandler
```

Qui cumulé avec des règles de rôles « isGranted » dans les Controller de notre choix ainsi que la configuration suivante :

```
login.html.twig | security.yaml | AccessDeniedHandler.php | doctrine.yaml ...\packages | TaskController.php | list.html.twig ...\task

1 <?php
2
3 namespace App\Security;
4
5 use Symfony\Component\HttpFoundation\Request;
6 use Symfony\Component\HttpFoundation\Response;
7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8 use Symfony\Component\Security\Core\Exception\AccessDeniedException;
9 use Symfony\Component\Security\Http\Authorization\AccessDeniedHandlerInterface;
10
11 class AccessDeniedHandler extends AbstractController implements AccessDeniedHandlerInterface
12 {
13     public function handle(Request $request, AccessDeniedException $accessDeniedException): ?Response
14     {
15         $this->addFlash('accessDenied', 'VOUS AVEZ ETE REDIRIGE SUR CETTE PAGE CAR : ' . $accessDeniedException->getMessage());
16         return $this->redirectToRoute('homepage');
17     }
18 }
```

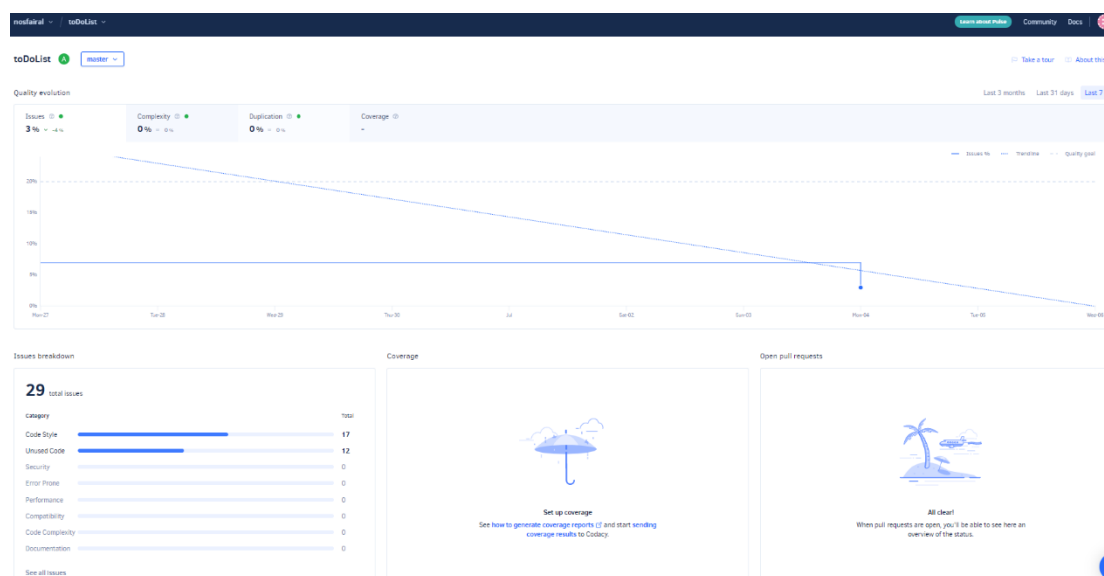
Du fichier « accessDeniedHandler.php » d’obtenir cette fonctionnalité.

Enfin un lien vers la page d’accueil a été réalisé dans la « navbar » du site au niveau du texte « To Do List app »

III. Audit de qualité

A) Analyse du code via « Codacy »

Actuellement on peut voir que notre code est de bonne qualité puisqu’il obtient une note de « A » sur codacy.



Repository name	Grade	Issues	Complexity	Duplication	Coverage	Last updated
toDoList	A	3%	0%	0%	-	4 hours ago
BilleMo-API	A	1%	0%	0%	-	a month ago
snowTricks	A	0%	0%	1%	-	2 months ago
postBlog	A	1%	0%	0%	-	4 months ago

B) Robustesse du code et norme PSR

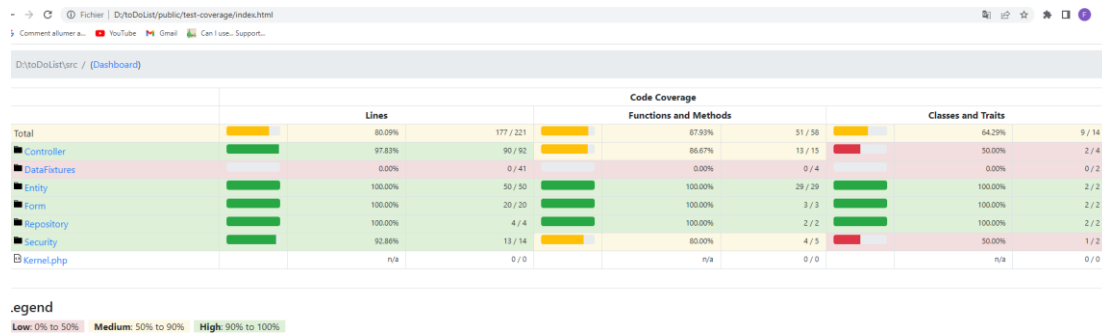
Afin de pallier le manque de tests unitaires et fonctionnels de base et de m'assurer de la robustesse de mon code, 48 tests avec 180 assertions ont été réalisés :

```
D:\toDoList>vendor\bin\phpunit
PHPUnit 9.0.0 by Sebastian Bergmann and contributors.

Testing
.....
Time: 00:24.003, Memory: 76.00 MB

OK (48 tests, 180 assertions)
```

J'ai aussi réalisé le rapport de couverture suivant avec un taux supérieur à 70 % (demande qui avait été faite par le client).



De plus je me suis assuré que le code de l'application respecte les PSR recommandés afin de proposer un code compréhensible et facilement évolutif. Dans ce sens un nettoyage en profondeur a été effectué à l'aide de code sniffer :

```
Git CMD
[1mPHPCBF RESULT SUMMARY[0m
-----
[1mFILE                                     FIXED  REMAINING[0m
-----
config\preload.php                        4      0
public\app_dev.php                       8      0
public\config.php                       20     13
public\index.php                         4      1
src\Controller\DefaultController.php     1      0
src\Controller\SecurityController.php     2      1
src\Controller\TaskController.php        10      0
src\Controller\UserController.php         8      3
src\DataFixtures\TaskFixtures.php        2      0
src\DataFixtures\UserFixtures.php        2      0
src\Entity\Task.php                      2      0
src\Entity\User.php                      2      0
src\Form\TaskType.php                    1      0
src\Form\UserType.php                   2      0
src\Repository\TaskRepository.php         1      0
src\Repository\UserRepository.php         1      0
src\Security\AccessDeniedHandler.php      2      1
src\Security\FormLoginAuthenticator.php   2      0
tests\appTest.php                        8      0
tests\bootstrap.php                      8      0
tests\Controller\DefaultControllerTest.php 2      0
tests\Controller\SecurityControllerTest.php 5      2
tests\Controller\TaskControllerTest.php   6      3
tests\Controller\UserControllerTest.php   2      3
tests\Entity\TaskTest.php                6      0
tests\Entity\UserTest.php                16      1
-----
[1mA TOTAL OF 127 ERRORS WERE FIXED IN 26 FILES[0m
-----
Time: 8.67 secs; Memory: 14MB
```

IV. Audit de performance (profiler de Symfony)

1. Analyse des résultats

Afin de s'assurer de la qualité de la performance du code de l'application j'ai réalisé une étude via le profiler de Symfony (Blackfire étant devenu payant pour toute pratique).

Parmi les routes de l'application :

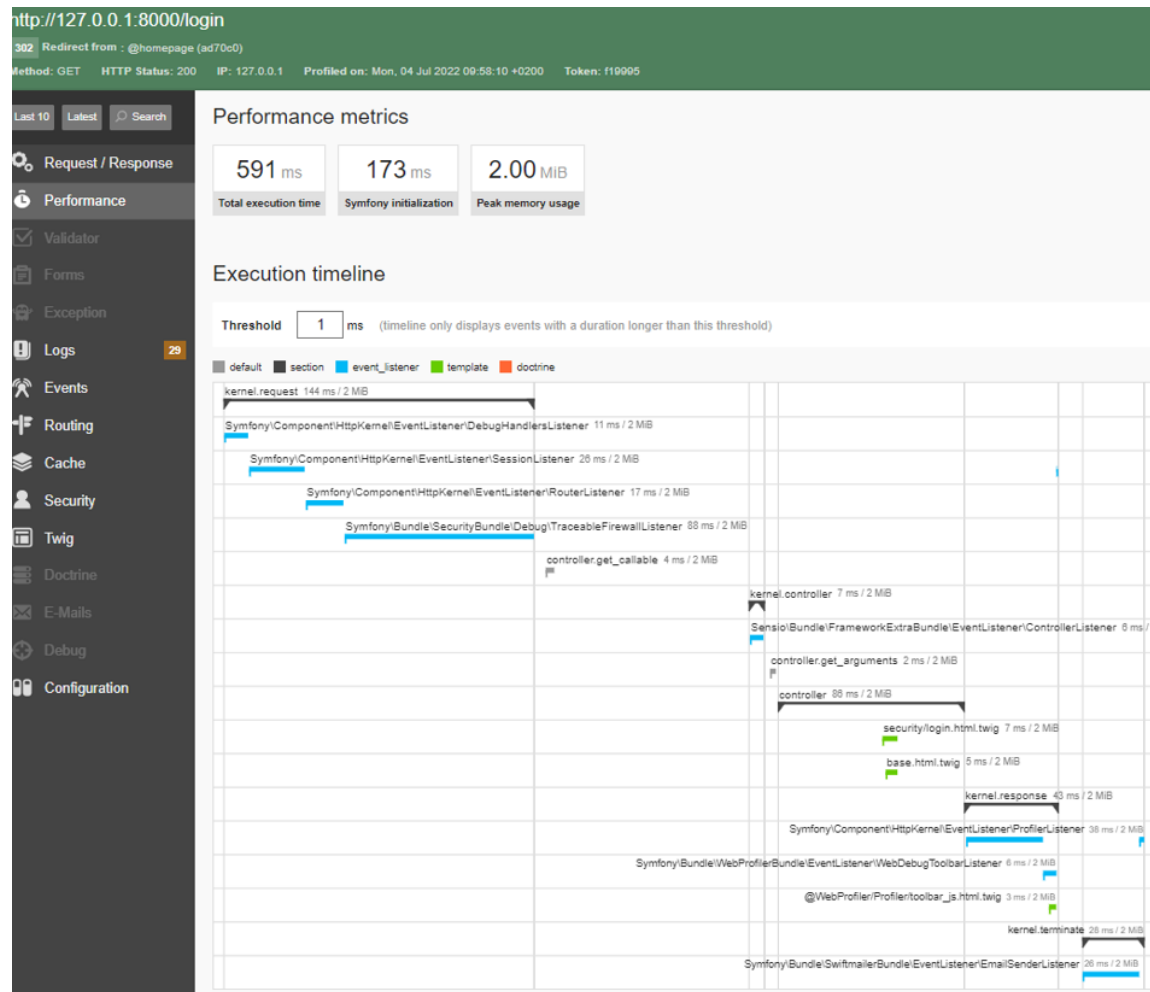
```
D:\todolistOrigin>php bin/console debug:route
```

Name	Method	Scheme	Host	Path
_wdt	ANY	ANY	ANY	/_wdt/{token}
_profiler_home	ANY	ANY	ANY	/_profiler/
_profiler_search	ANY	ANY	ANY	/_profiler/search
_profiler_search_bar	ANY	ANY	ANY	/_profiler/search_bar
_profiler_phpinfo	ANY	ANY	ANY	/_profiler/phpinfo
_profiler_search_results	ANY	ANY	ANY	/_profiler/{token}/search/results
_profiler_open_file	ANY	ANY	ANY	/_profiler/open
_profiler	ANY	ANY	ANY	/_profiler/{token}
_profiler_router	ANY	ANY	ANY	/_profiler/{token}/router
_profiler_exception	ANY	ANY	ANY	/_profiler/{token}/exception
_profiler_exception_css	ANY	ANY	ANY	/_profiler/{token}/exception.css
_twig_error_test	ANY	ANY	ANY	/_error/{code}.{_format}
homepage	ANY	ANY	ANY	/
login	ANY	ANY	ANY	/login
login_check	ANY	ANY	ANY	/login_check
logout	ANY	ANY	ANY	/logout
task_list	ANY	ANY	ANY	/tasks
task_create	ANY	ANY	ANY	/tasks/create
task_edit	ANY	ANY	ANY	/tasks/{id}/edit
task_toggle	ANY	ANY	ANY	/tasks/{id}/toggle
task_delete	ANY	ANY	ANY	/tasks/{id}/delete
user_list	ANY	ANY	ANY	/users
user_create	ANY	ANY	ANY	/users/create
user_edit	ANY	ANY	ANY	/users/{id}/edit

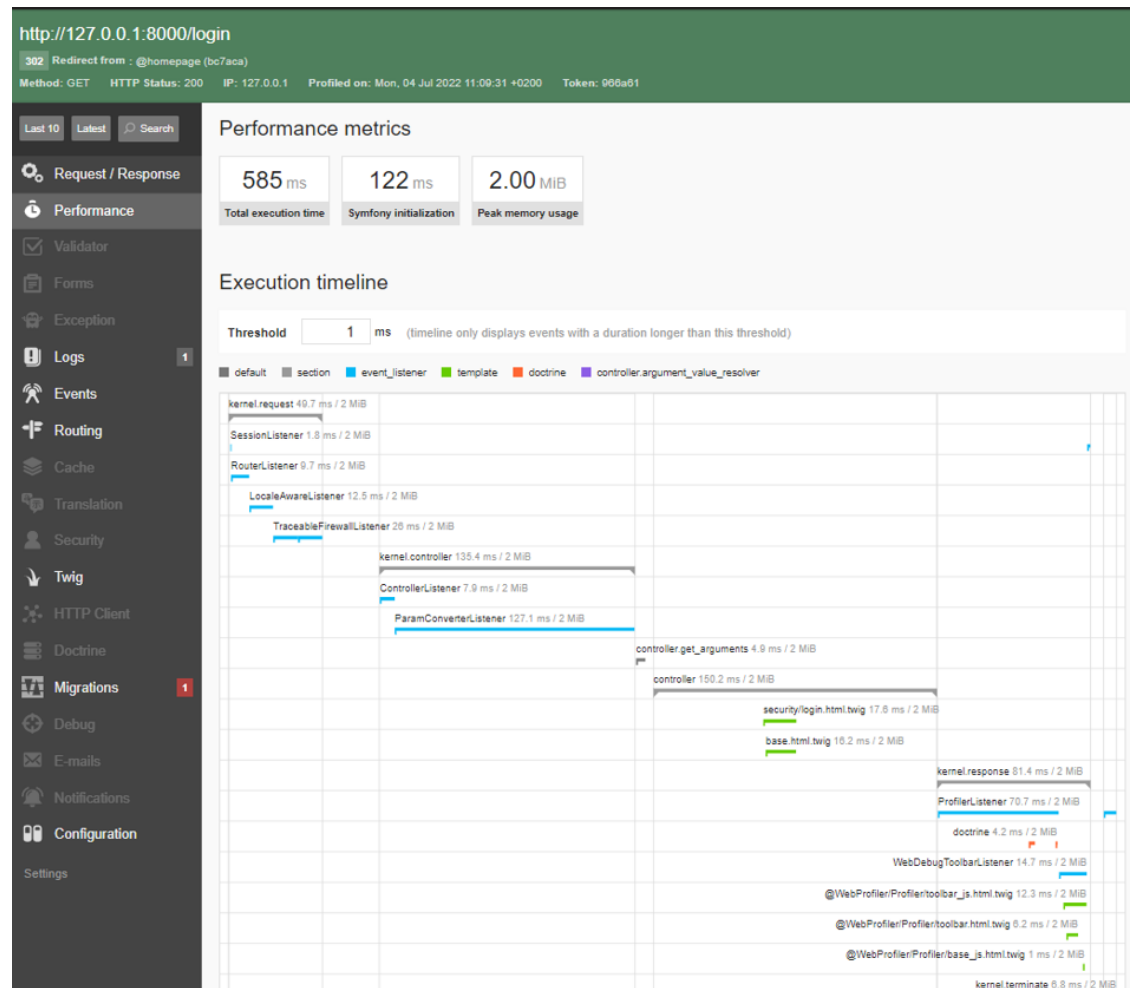
J'ai effectué des tests sur les principales routes communes à l'application avant et après modifications :

✓ Route « /login » :

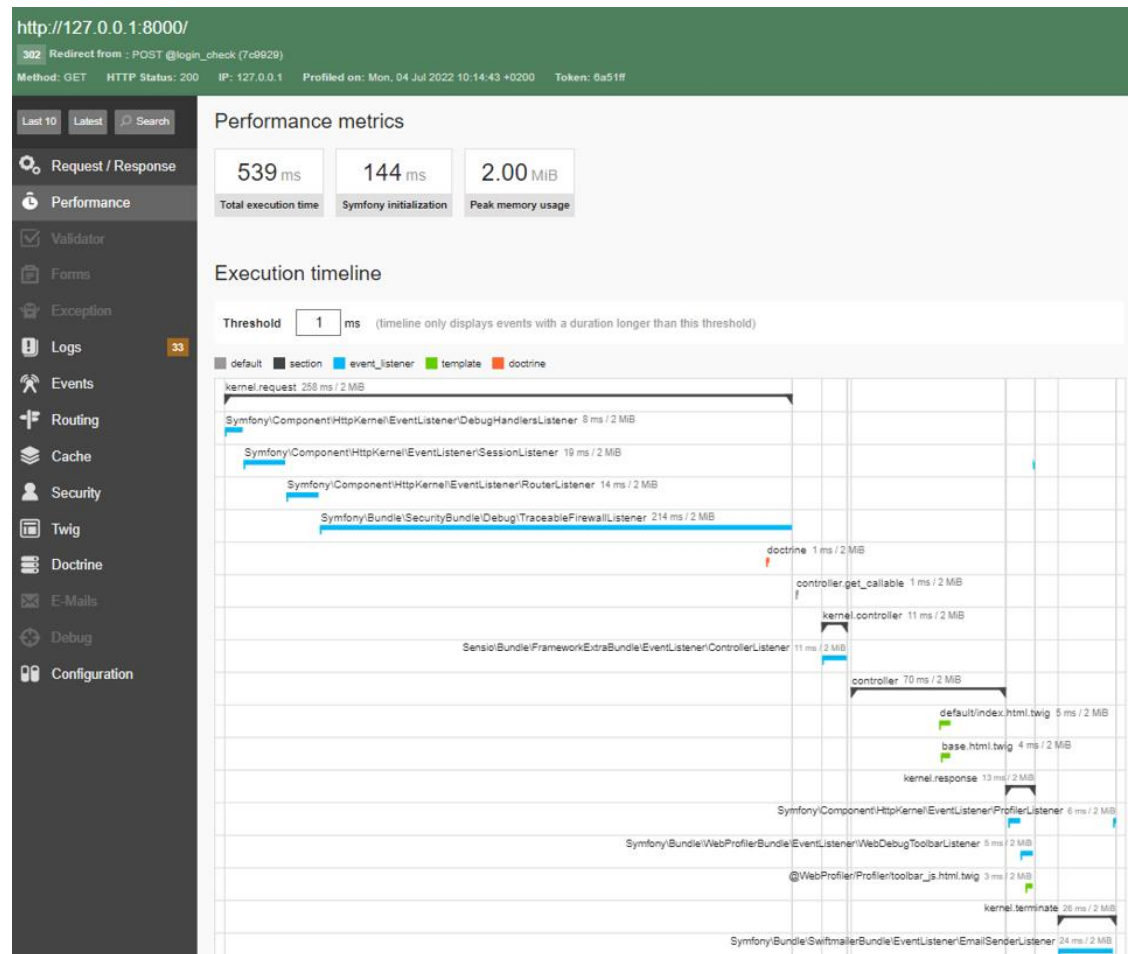
Version 3.4 :



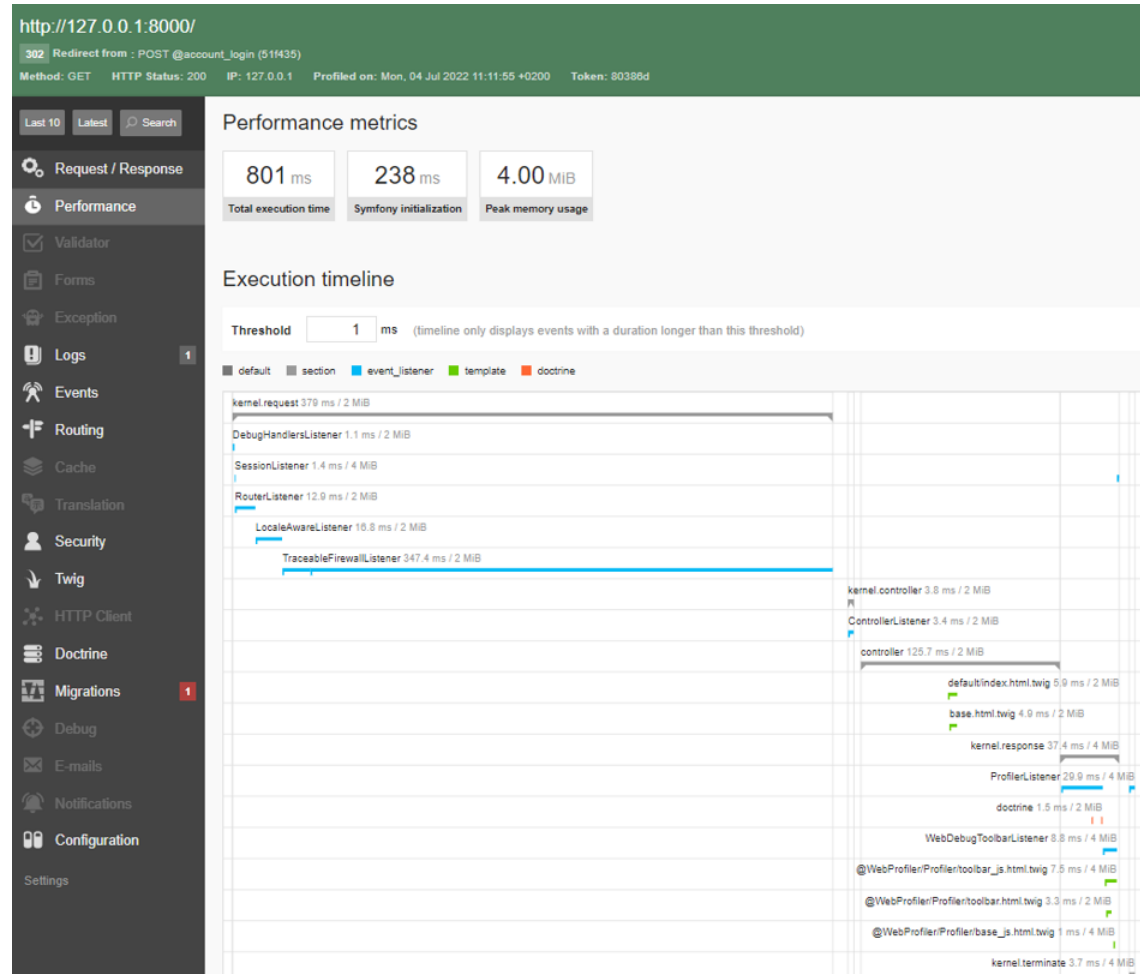
Version 5.4 :



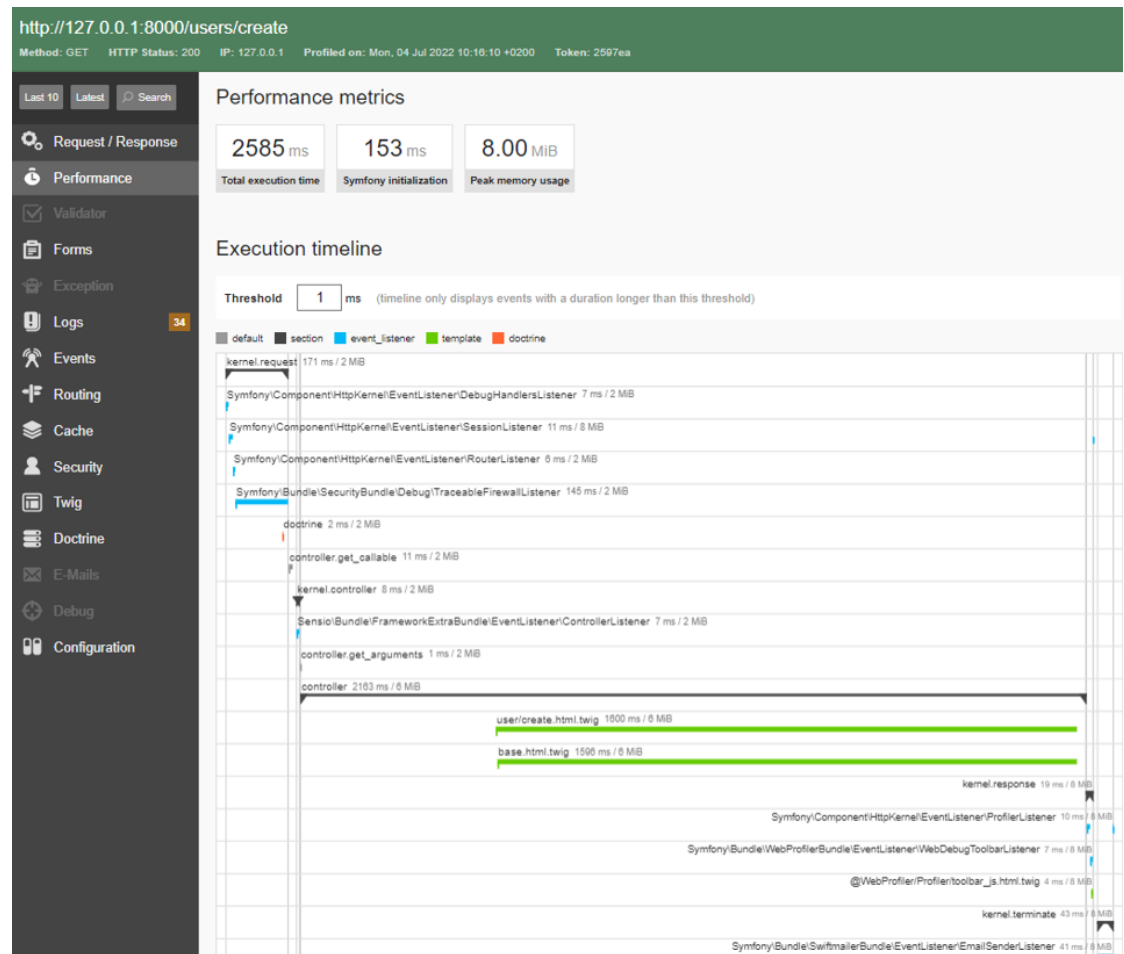
- ✓ Route « / » après redirection depuis login
- Version 3.4 :



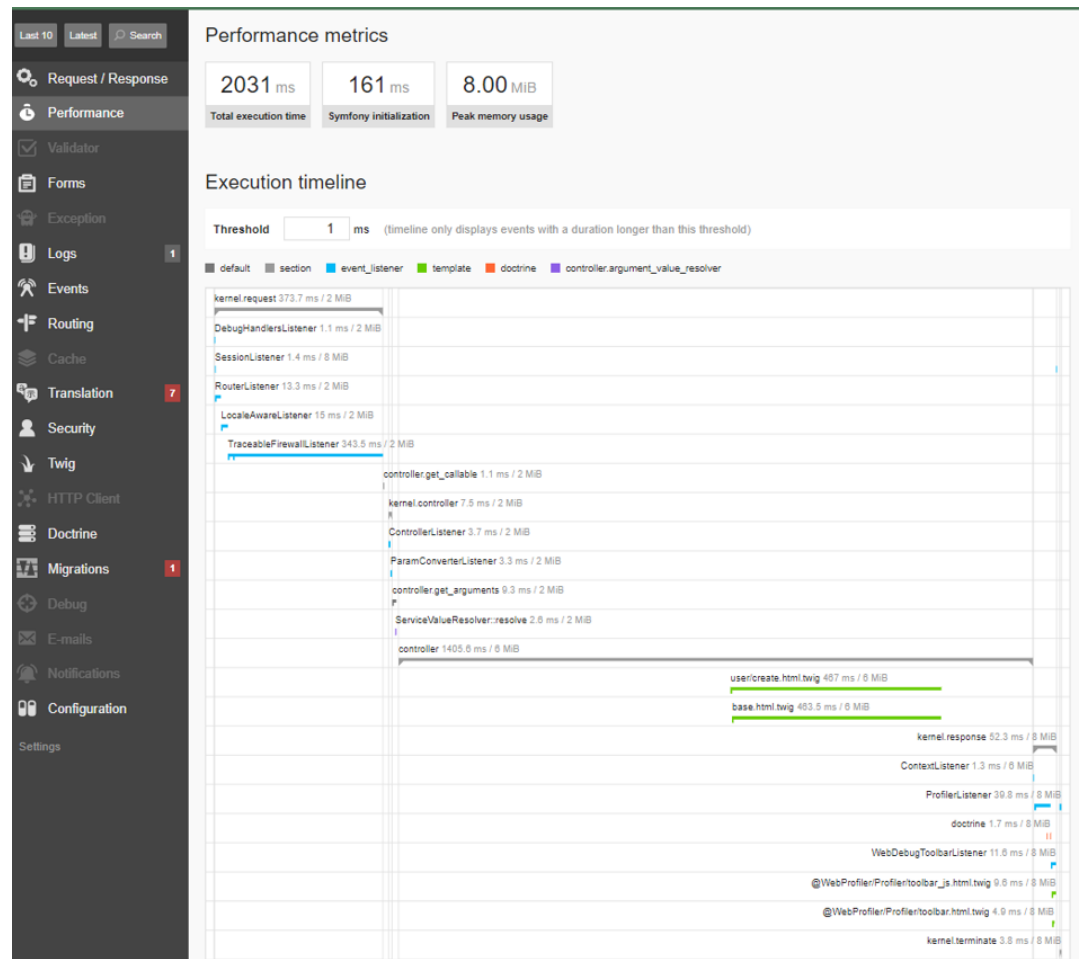
Version 5.4 :



- ✓ Route « /users/create » pour créer utilisateur :
Version 3.4 :

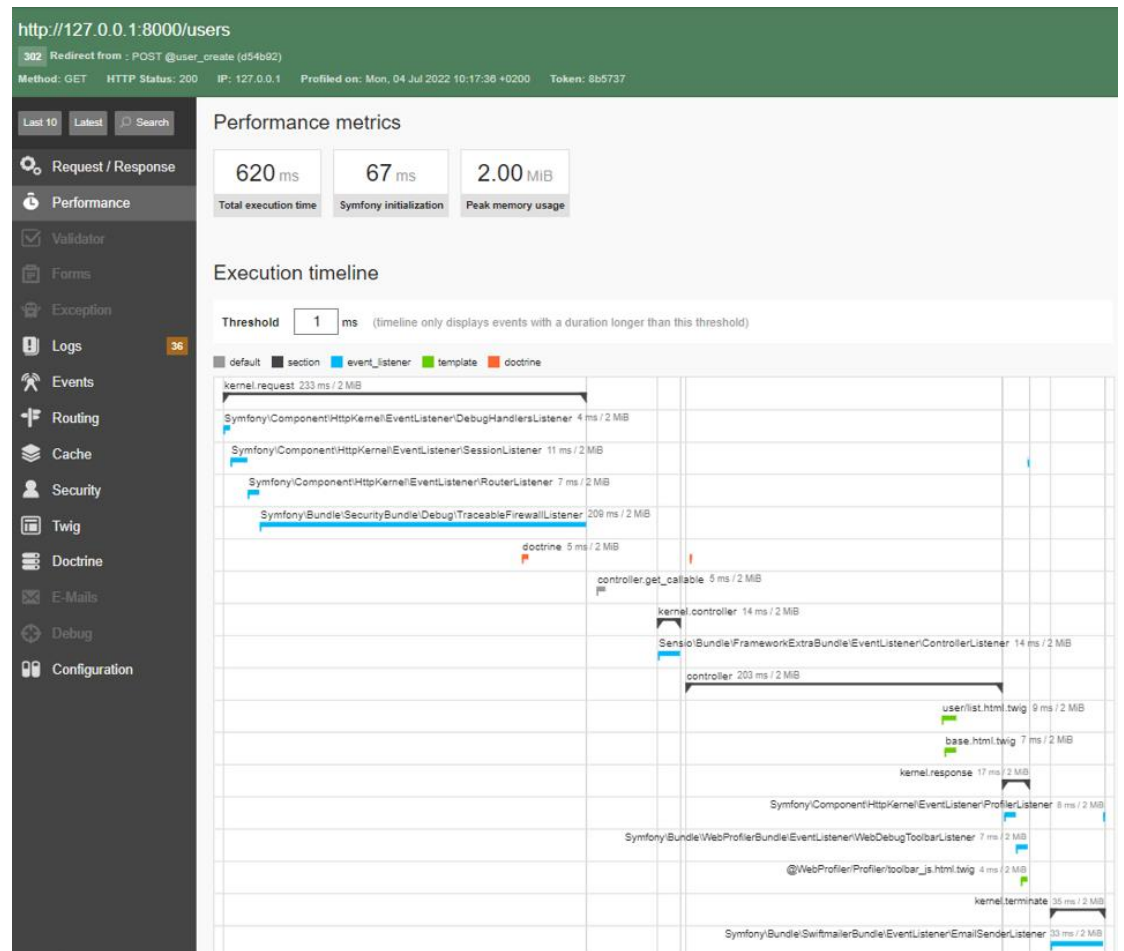


Version 5.4 :

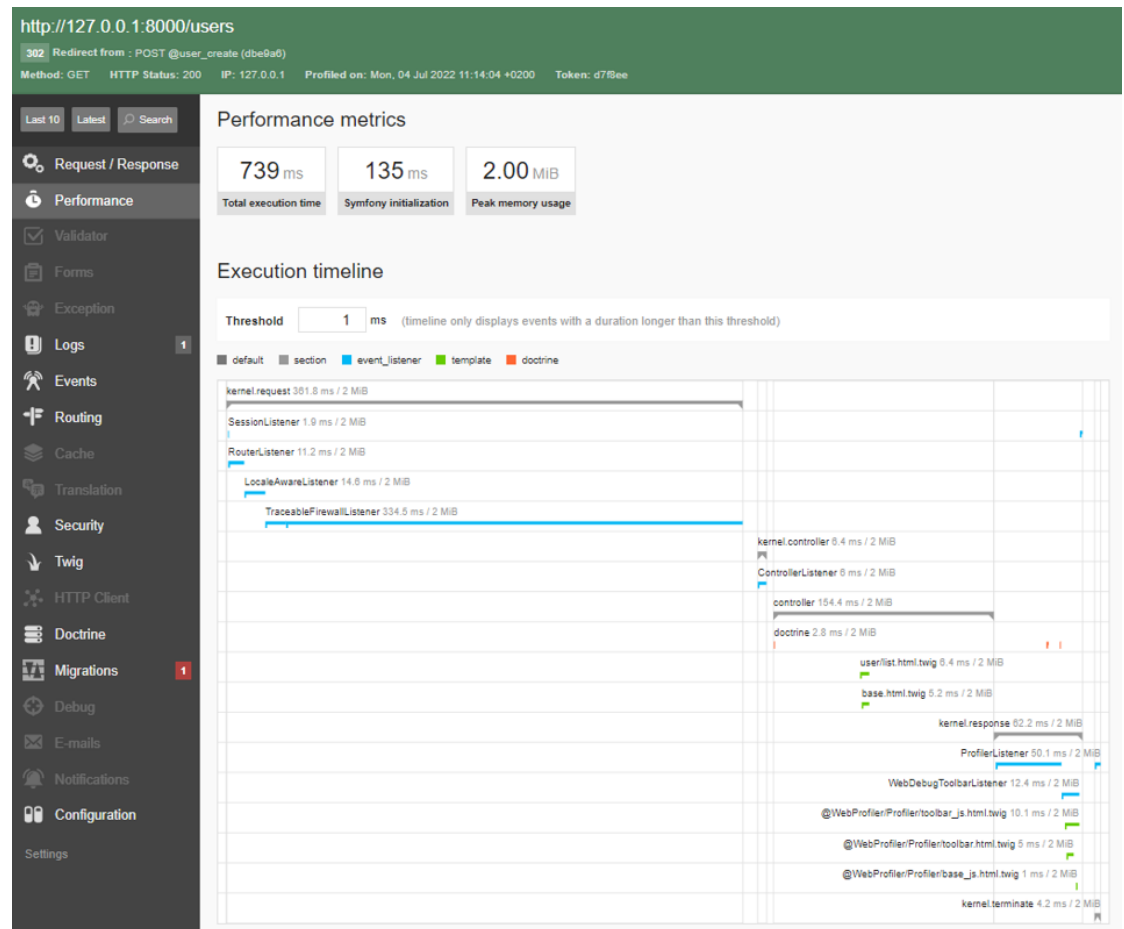


- ✓ Route « /users » redirection après création de l'utilisateur dans le cas d'un administrateur pour la nouvelle version.

Version 3.4 :

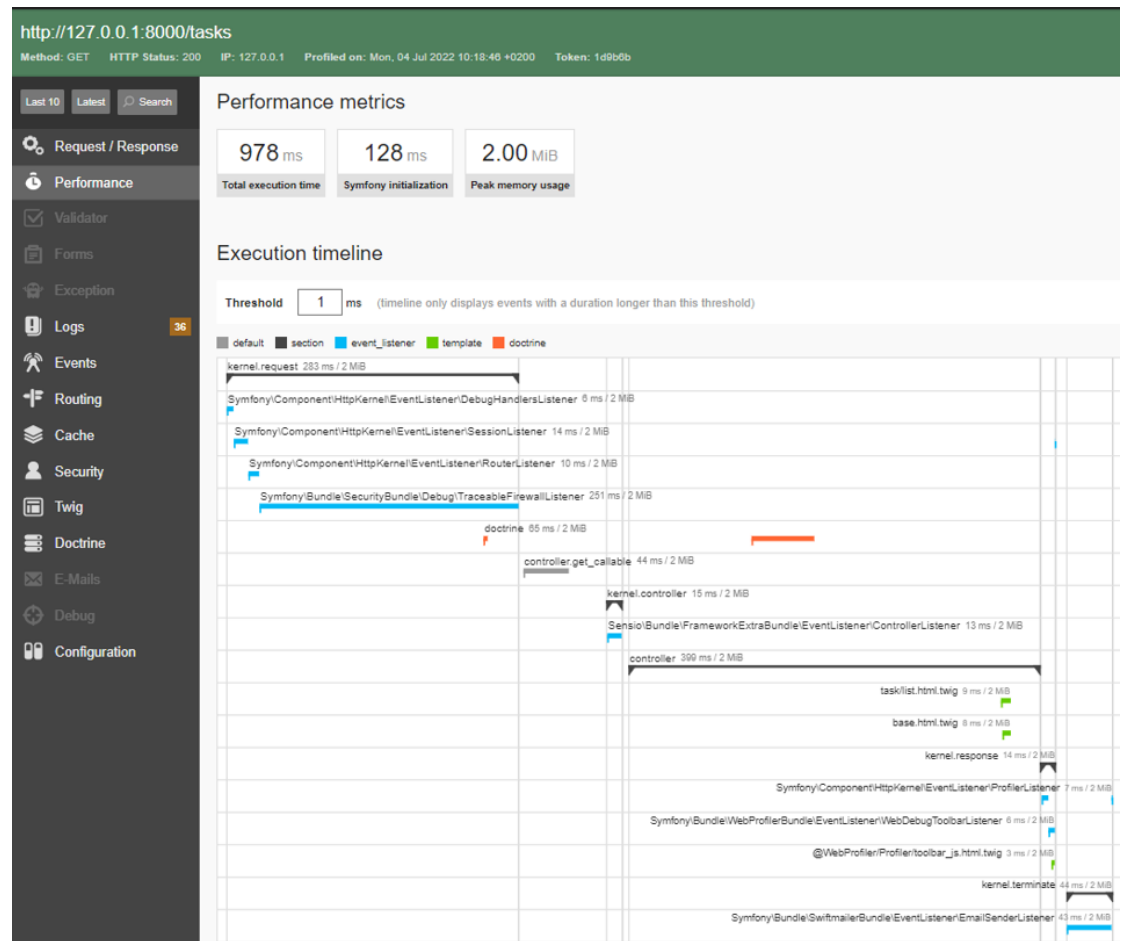


Version 5.4 :

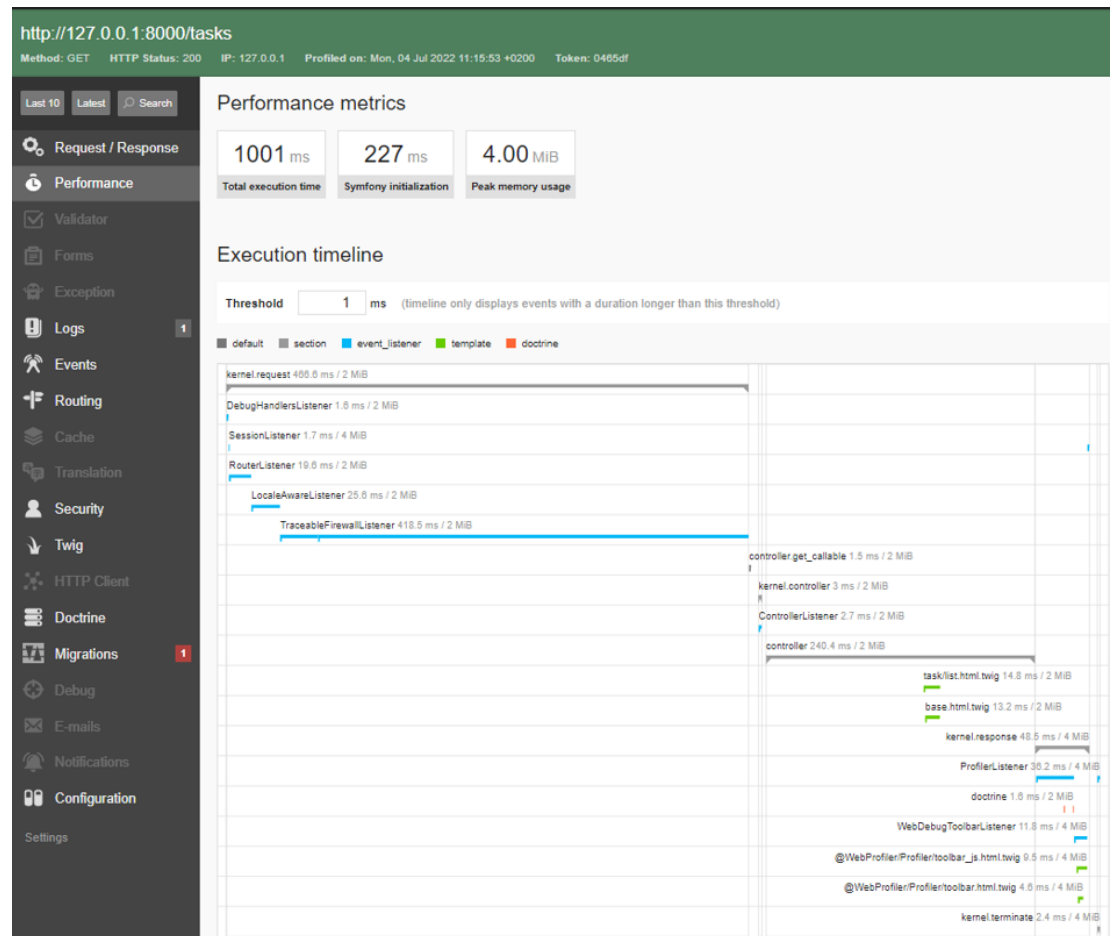


- ✓ Route « /tasks » liste des taches indistinctes pour l'ancienne version, juste les non terminées pour la nouvelle.

Version 3.4 :

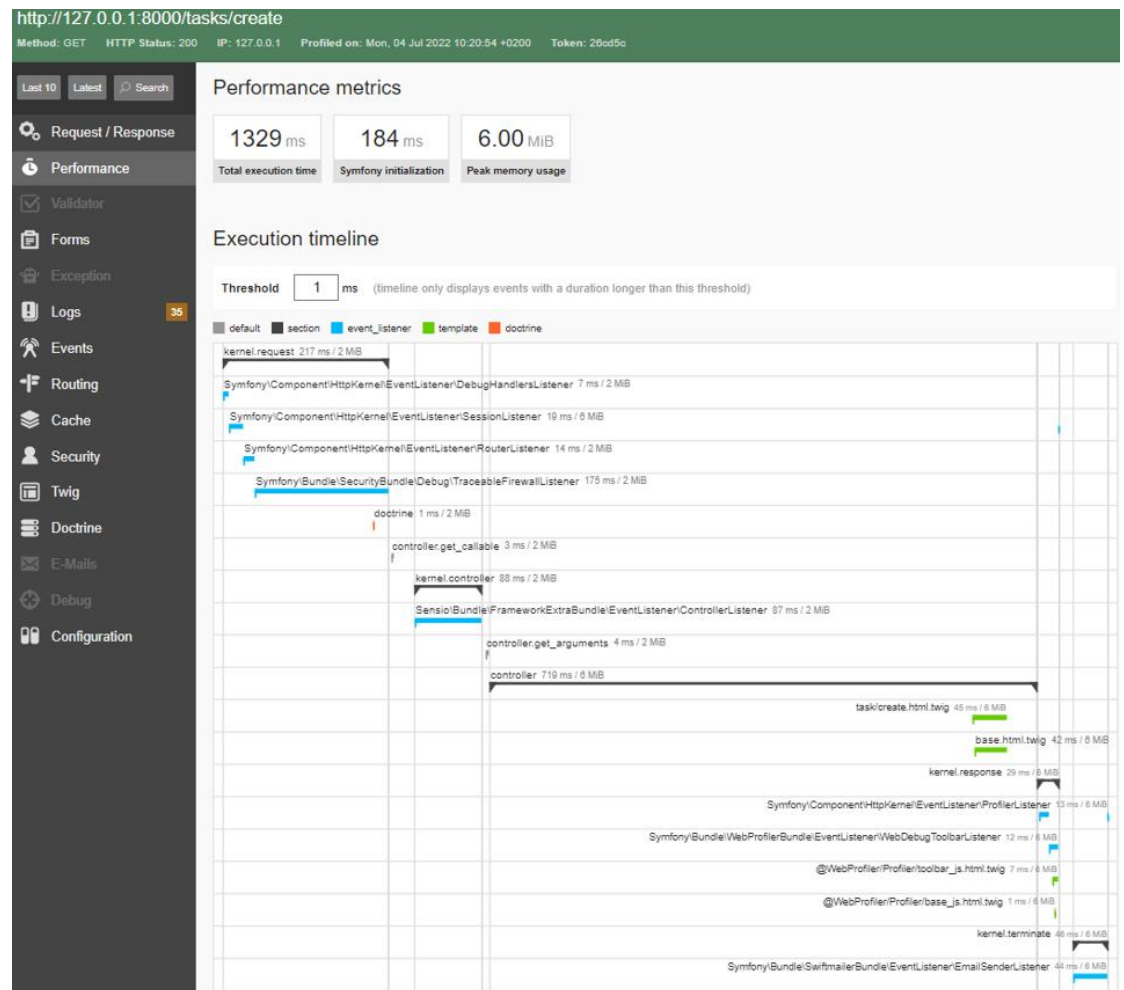


Version 5.4 :

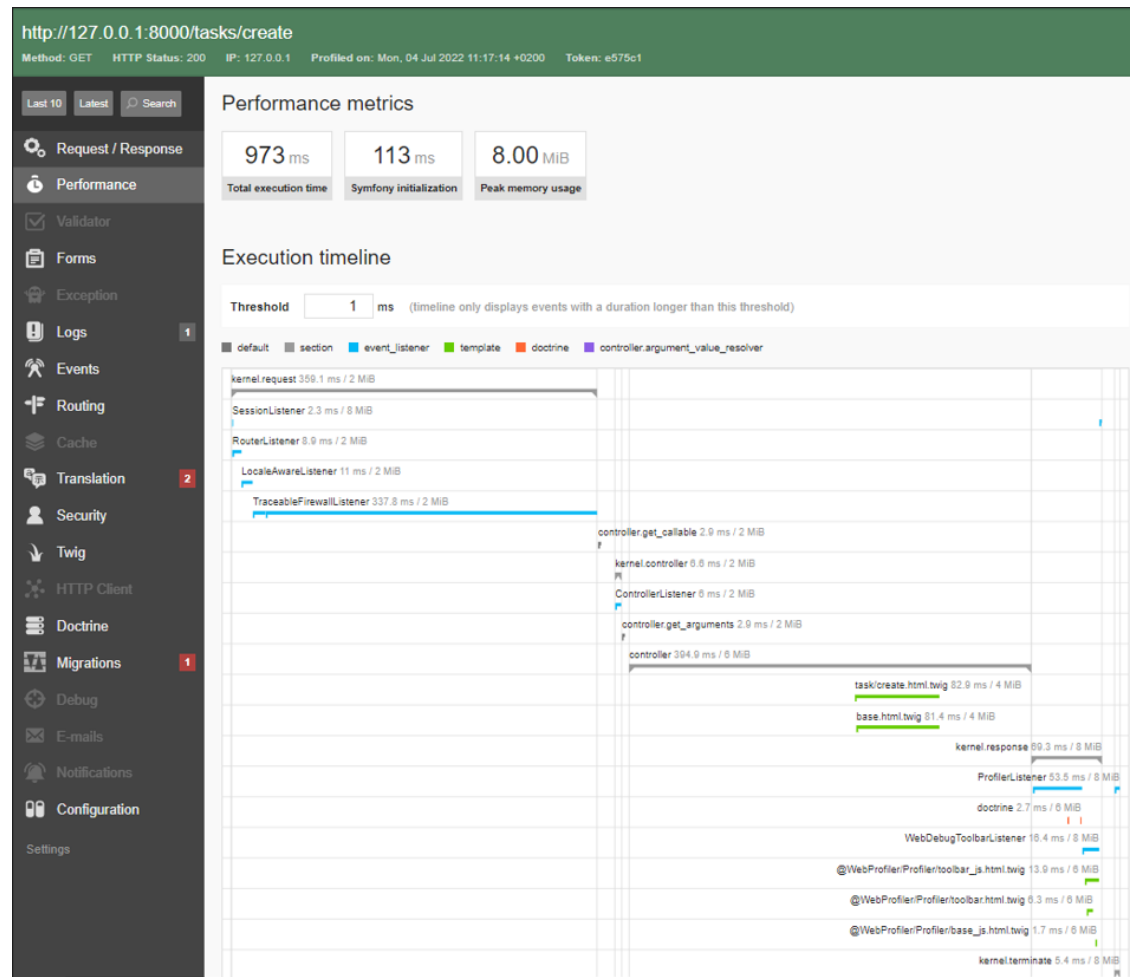


- ✓ Route « /tasks/create » Création d'une tâche.

Version 3.4 :



Version 5.4 :



Au vu des résultats ci-dessus nous pouvons constater :

- ✓ Qu'en passant de la version 3.4 à la 5.4 l'application gagne globalement en vitesse dès lors que la sécurité renforcée sur la nouvelle mouture n'intervient pas.
- ✓ Cela affecte peu la mémoire utilisée.

V. Axes d'améliorations possibles

1. PLAN D'AMELIORATION DES PERFORMANCES

- Optimisation de l'autoloader au niveau 2/B

<https://getcomposer.org/doc/articles/autoloader-optimization.md>

- Utiliser le système de cache de Doctrine et Symfony

- Utilisation de l'AJAX pour éviter de recharger toute une page à chaque Interaction (exemple : lors du marquage d'une tâche comme faite)
- Implémenter un système de lazy loading pour les images
- Ne pas intégrer les DataFixtures avant la mise en production
- Ne pas intégrer les tests avant la mise en production
- Upgrade de version Symfony et PHP

Ici la documentation officielle de Symfony pour améliorer les performances :

<https://symfony.com/doc/5.4/performance.html>

2.AJOUT DE FONCTIONNALITES POSSIBLES

- Supprimer un utilisateur par l'administrateur
- Informer une date de fin pour chaque tâche
- Rappel par mail si une tâche dépasse le délai imparti ou arrive à échéance
- Message de confirmation avant de supprimer une tâche
- Section mot de passe oublié pour les utilisateurs
- Section page de profil pour que les utilisateurs puissent modifier leurs

données personnelles

3.PLAN D'AMELIORATION DU CODE

- Fonction de vérification que l'utilisateur est administrateur dans un service si une nouvelle fonctionnalité demande de le vérifier
- Mettre en place des outils d'intégration continu comme GitHub Actions :

<https://github.com/actions>

- Vérifier la qualité du code si ajout de nouvelles fonctionnalités (avec Codacy par exemple) ou modification de fonctionnalités existantes

- Respecter les bonnes pratiques :

- PSR-1 : <https://www.php-fig.org/psr/psr-1/>

- PSR-2 : <https://www.php-fig.org/psr/psr-2/>

- Symfony standard : <https://symfony.com/doc/5.4/contributing/code/standards.html>