

Audit de performances

Audit de performances.....	1
Contexte.....	2
Outils d'analyse.....	2
Analyse de l'application sur sa version initiale.....	2
Rapport de qualité de code (Codeclimate).....	3
Rapport de performance.....	4
Améliorations envisagées et solutions à mettre en œuvre.....	4
Architecture.....	4
Revue de code.....	5
Analyse de l'application sur sa version finale.....	5
Rapport de qualité de code (Codeclimate).....	6
Rapport de performance.....	7
Comparaison des 2 applications.....	7
Conclusion.....	8
Captures d'écrans des données métriques du profiler Symfony.....	9
Symfony 3.1.....	9
Symfony 6.4.....	25

Contexte

L'application Todo List, développée par la startup ToDo & Co, offre aux utilisateurs enregistrés la possibilité de gérer leurs tâches quotidiennes. Initiée dans le cadre d'une stratégie de développement axée sur le MVP (Minimum Viable Product), cette approche a permis à la startup de présenter rapidement l'application et de recueillir les retours des investisseurs potentiels. Cependant, cette étape n'est que le début, car ToDo & Co s'engage désormais à améliorer la qualité de l'application en ajoutant de nouvelles fonctionnalités, en mettant en place des tests automatisés et en corrigeant les anomalies.

Avant de se plonger dans les modifications du code de l'application, une analyse approfondie du projet initial est nécessaire, se concentrant sur la qualité du code et les performances de l'application. Cette étape est cruciale pour identifier les éventuels besoins de mise à jour visant à optimiser le développement et l'expérience utilisateur pour les futurs clients.

Ce document présente donc un rapport détaillé comprenant :

- Une analyse de l'application dans sa version initiale.
- Des pistes d'amélioration possibles.
- Une analyse de l'application dans sa version actuelle.

Outils d'analyse

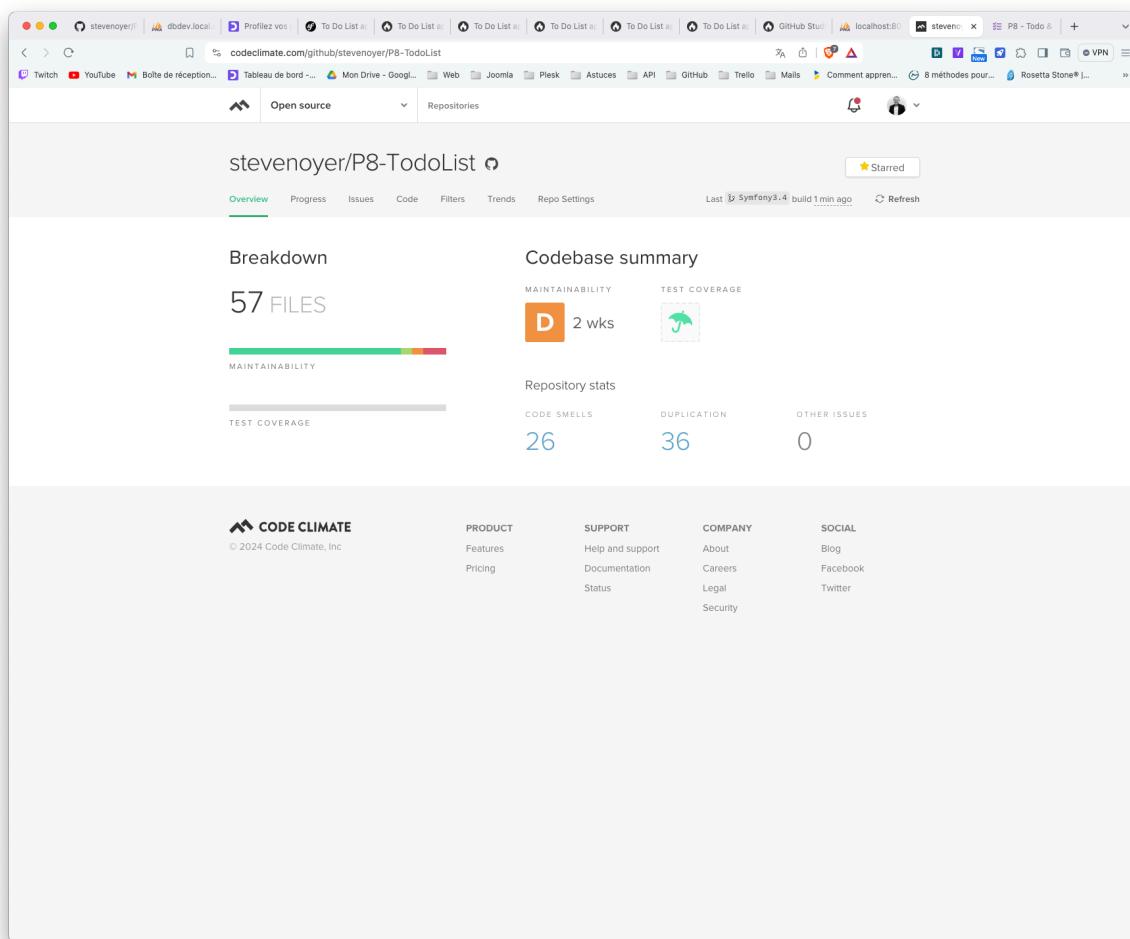
Les outils d'analyse utilisés pour réaliser cet audit sont listés ci-dessous :

- Codeclimate (qualité de code)
- Profiler de Symfony (performances)

Analyse de l'application sur sa version initiale

- Version : Symfony 3.1
- PHP 5.6
- Environnement de dev : WAMP & Symfony CLI

Rapport de qualité de code (Codeclimate)



Rapport de performance

Symfony 3				
Routes	Total execution time (ms)	Symfony initialization (ms)	Peak memory usage (MB)	
Login	449	176	14,25	
Login after disconnect	381	133	14,25	
Register	1767	161	23,75	
Home	787	286	18,75	
Tasks	607	157	19,5	
Task create	1430	427	20,75	
Task edit	1436	431	21	
Tasks after change status	460	134	17	
Tasks after create	525	163	17	
Tasks after edit	523	160	17	
Tasks after delete	511	163	17	
Users	451	136	17	
Create user	515	136	20,5	
Edit user	550	144	20,5	
Users after create	520	193	17	
Users after edit	426	136	17	
Moyenne	708,63	196,00	18,27	

Améliorations envisagées et solutions à mettre en œuvre

Architecture

La première étape a consisté à migrer le projet vers une version LTS du framework.

La version initiale du projet était sur Symfony 3.1, qui n'est plus maintenue depuis janvier 2017. Il était donc nécessaire de mettre à jour la version à Symfony 6.4, qui est une version LTS depuis novembre 2023. Le support de Symfony 6.4 en matière de bugs se terminera en novembre 2026 et deviendra obsolète en novembre 2027, date à laquelle le support de correction de sécurité s'arrêtera.

La version de PHP a également été mise à jour, passant de la version 5.4 à la 8.3. La version 8.3 sera supportée jusqu'au 23 novembre 2026 avant de passer à la 8.4. Les versions inférieures à la version 8.1 ne sont donc pas recommandées en raison de problèmes de sécurité et de vulnérabilité.

La mise à jour du framework s'est effectuée en plusieurs étapes pour résoudre de nombreuses erreurs (obsolètes) et adapter l'architecture. Cette mise à jour a également inclus l'actualisation de toutes les bibliothèques PHP utilisées.

Versions Symfony	Versions PHP
3.1	5.5.9
3.4	5.5.9
4.0	7.1
4.4	7.1
5.0	7.4
5.4	7.4
6.0	8.1
6.3	8.3
6.4	8.3

Environnement de travail

Chaque développeur a la liberté de choisir son environnement de travail. La base de données peut être mise en place via un logiciel comme WAMP/MAMP, une base de données externe, ou en utilisant Docker, dont la configuration se trouve dans le fichier `compose.override.yaml` à la racine du projet.

Revue de code

Avant chaque commit, veuillez effectuer des tests unitaires sur chaque nouvelle fonctionnalité ou correction de bug à l'aide de PHPUnit. Assurez-vous que la qualité de votre code est excellente avant chaque pull-request en utilisant l'outil Codeclimate.

Analyse de l'application sur sa version finale

- Version : Symfony 6.4
- PHP 8.3
- Environnement de dev : Docker & Symfony CLI

Rapport de qualité de code (Codeclimate)

The screenshot shows the CodeClimate analysis interface for the repository `stevenoyer/P8-TodoList`. The main dashboard includes a breakdown of 93 files, a maintainability score of A (5 hrs), and repository stats showing 2 code smells, 0 duplication, and 0 other issues. The footer contains links to Code Climate's product, support, company, and social media pages.

Breakdown
93 FILES

Maintainability
A 5 hrs

Codebase summary

Repository stats

CODE SMELLS	DUPLICATION	OTHER ISSUES
2	0	0

CODE CLIMATE
© 2024 Code Climate, Inc.

PRODUCT	SUPPORT	COMPANY	SOCIAL
Features	Help and support	About	Blog
Pricing	Documentation	Careers	Facebook
	Status	Legal	Twitter
		Security	

Rapport de performance

Symfony 6				
Routes	Total execution time (ms)	Symfony initialization (ms)	Peak memory usage (MB)	
Login	21	9	4	
Login after disconnect	8	2	6	
Login after register	9	2	6	
Register	36	12	6	
Home	14	2	4	
Home after login	16	2	6	
Tasks	58	15	6	
Task create	39	4	6	
Task edit	30	8	6	
Tasks after change status	17	3	6	
Tasks after create	25	3	6	
Tasks after edit	21	3	6	
Tasks after delete	17	2	6	
Users	17	2	4	
User create	44	8	6	
User edit	57	17	6	
Users after create	15	2	6	
Users after edit	18	2	6	
Moyenne	25,67	5,44	5,67	

Comparaison des 2 applications



Nous pouvons remarquer qu'en passant l'application à Symfony 6.4 + PHP 8.3, nous constatons une nette amélioration dans le temps d'exécution total du code, l'initialisation de Symfony, et le pic d'utilisation de la mémoire. Cela constitue une différence notable avec :

- une différence de **682,96 ms** dans le temps d'exécution du code.
- une différence de **190,56 ms** dans l'initialisation du framework Symfony.
- une différence de **12,6 MB** dans le pic d'utilisation de la mémoire.

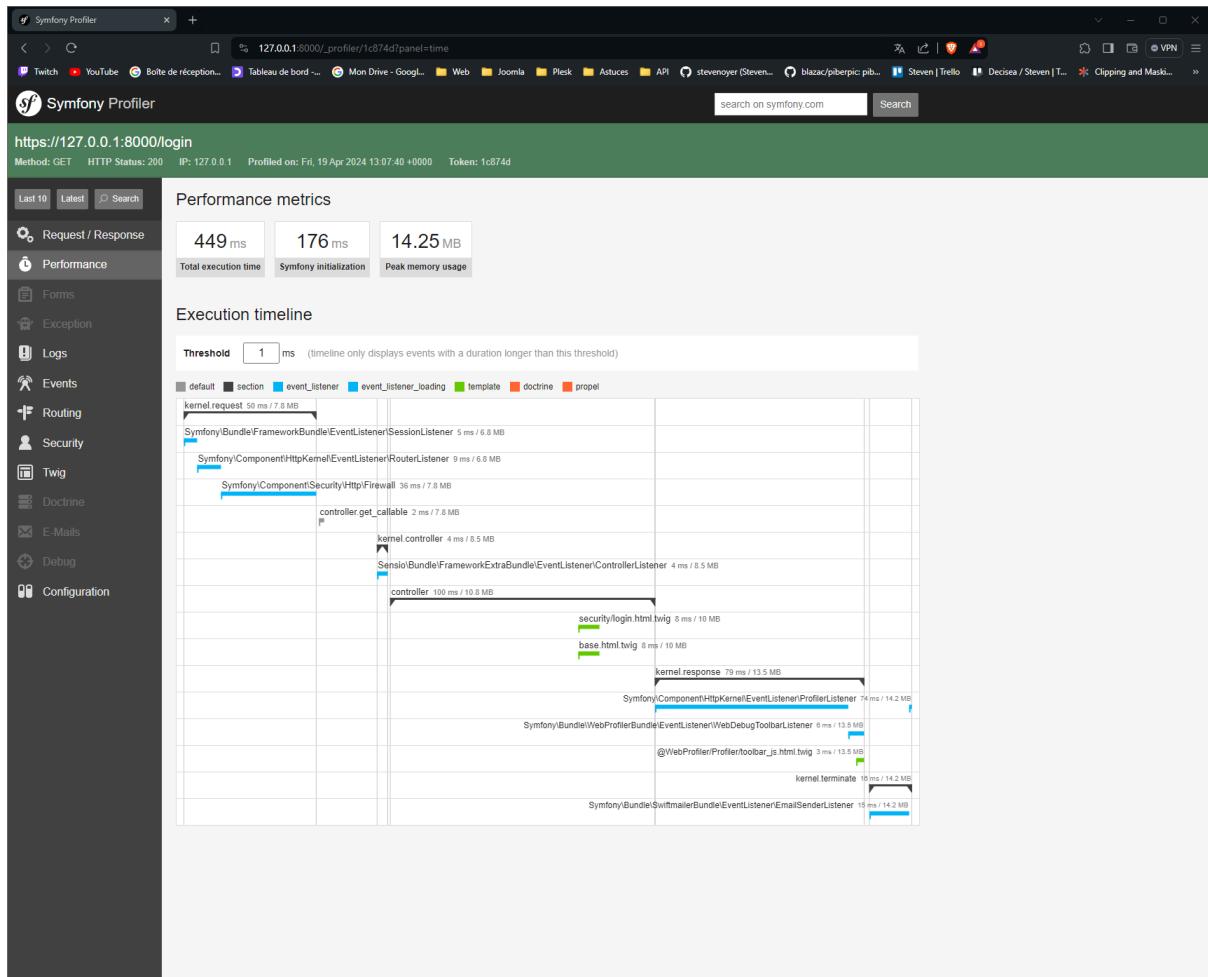
Conclusion

- La migration vers la version LTS du framework Symfony garantit la compatibilité avec les versions mineures à venir et offre un support pour les trois prochaines années avant la sortie de sa prochaine version majeure.
- La mise en place des outils d'analyse est essentielle pour tester les fonctionnalités et garantir un code de qualité.
- Le profiler de Symfony nous a permis d'analyser en détail les performances de l'application grâce aux données métriques fournies. Avec cette analyse, nous avons pu déterminer les améliorations en temps d'exécution du code, le temps d'initialisation du framework et la réduction du pic d'utilisation de la mémoire. Pour des analyses plus approfondies à l'avenir, Todo&Co peut envisager de souscrire à un abonnement à BlackFire ou New Relic dans le cadre d'une réflexion sur l'intégration continue. L'avantage de ces outils est qu'ils sont spécialisés dans l'analyse des performances et automatisent le processus d'analyse avec des métriques plus poussées.

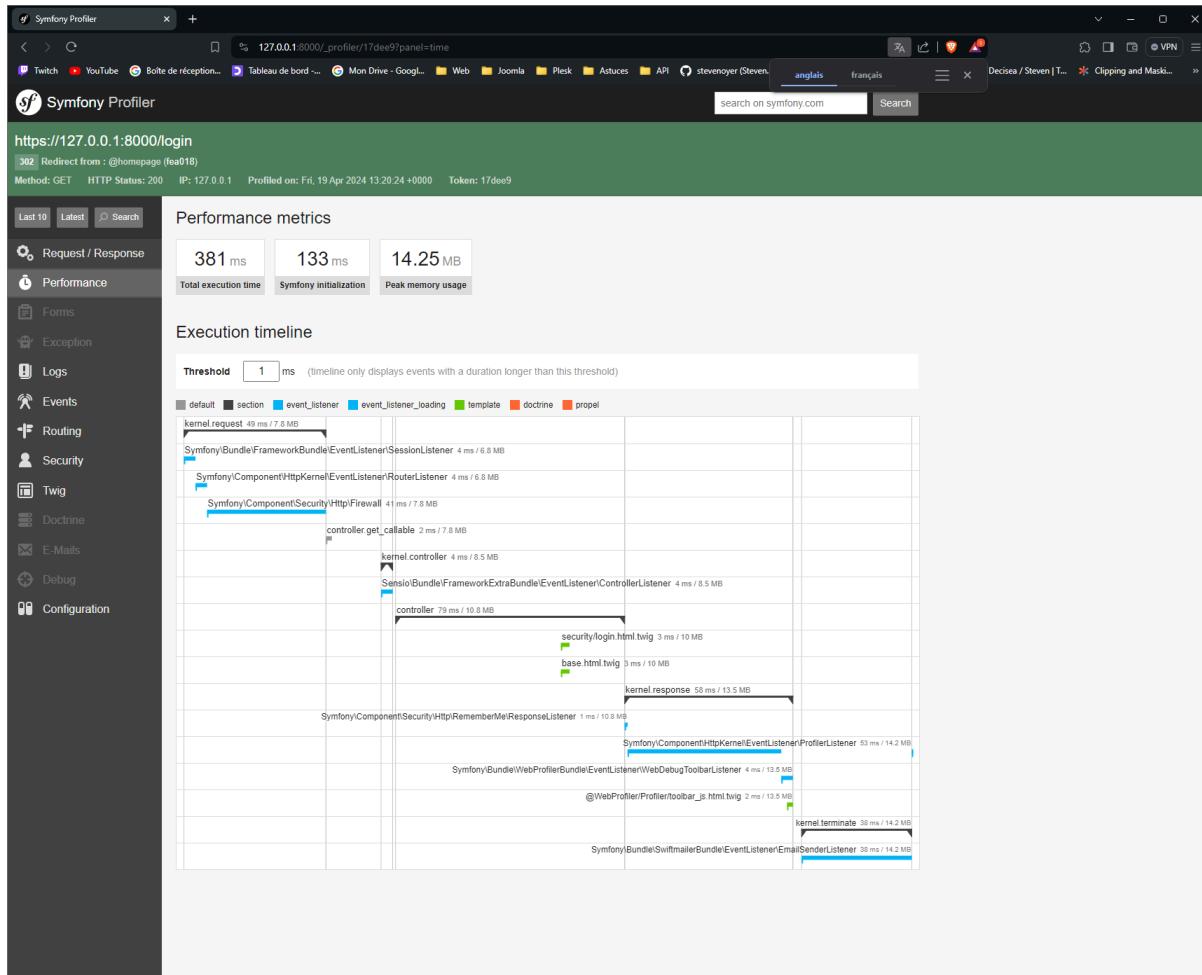
Captures d'écrans des données métriques du profiler Symfony

Symfony 3.1

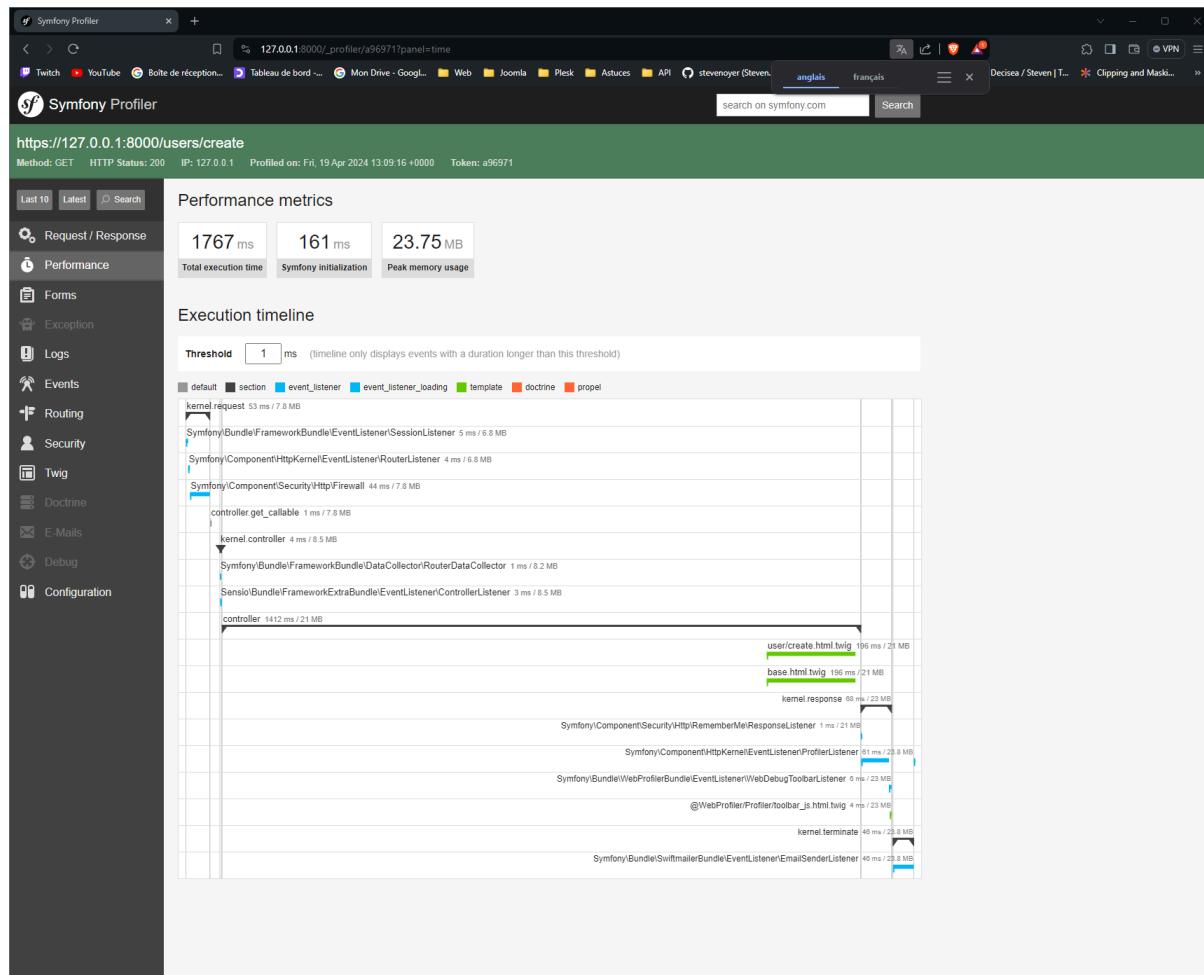
Login



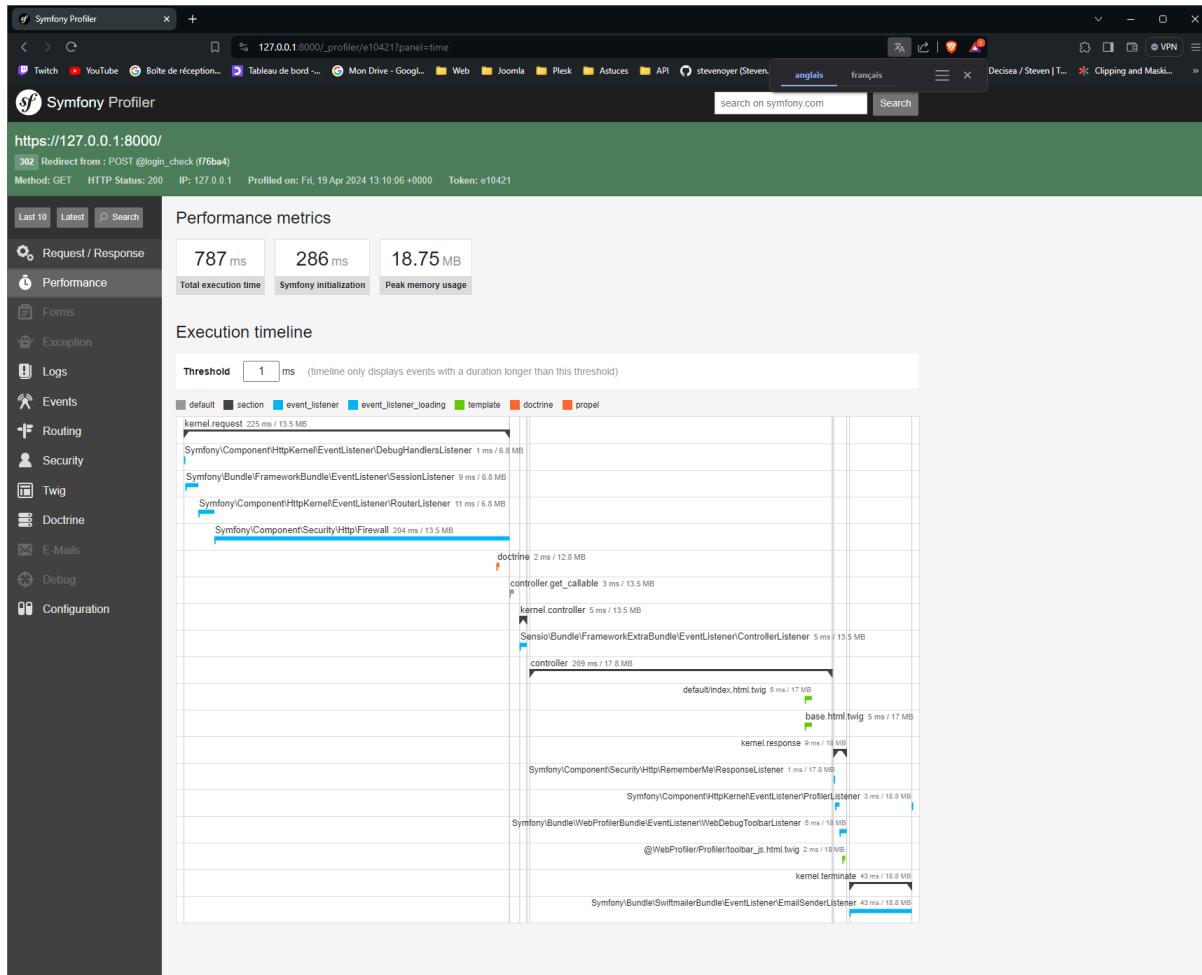
Login after disconnect



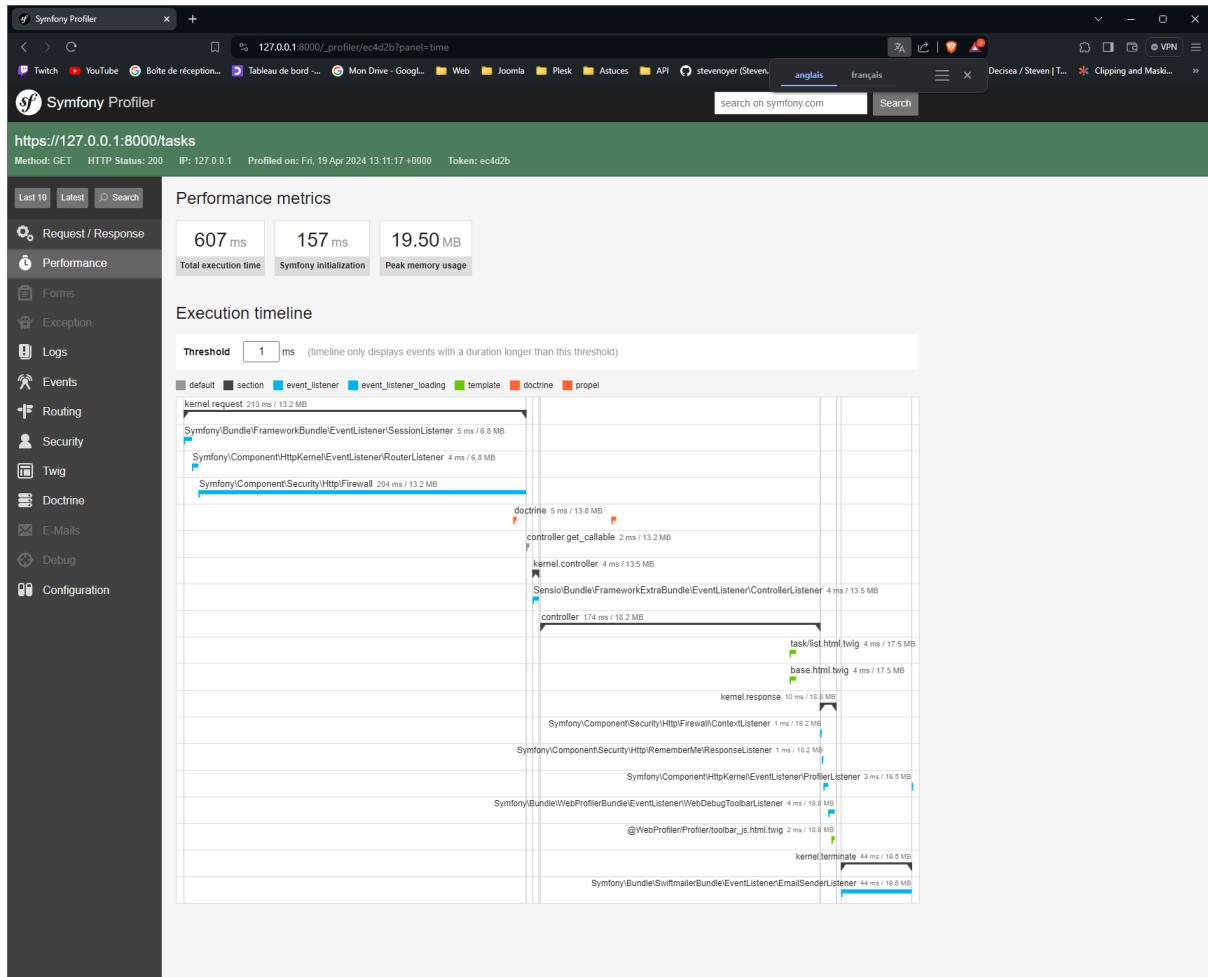
Register



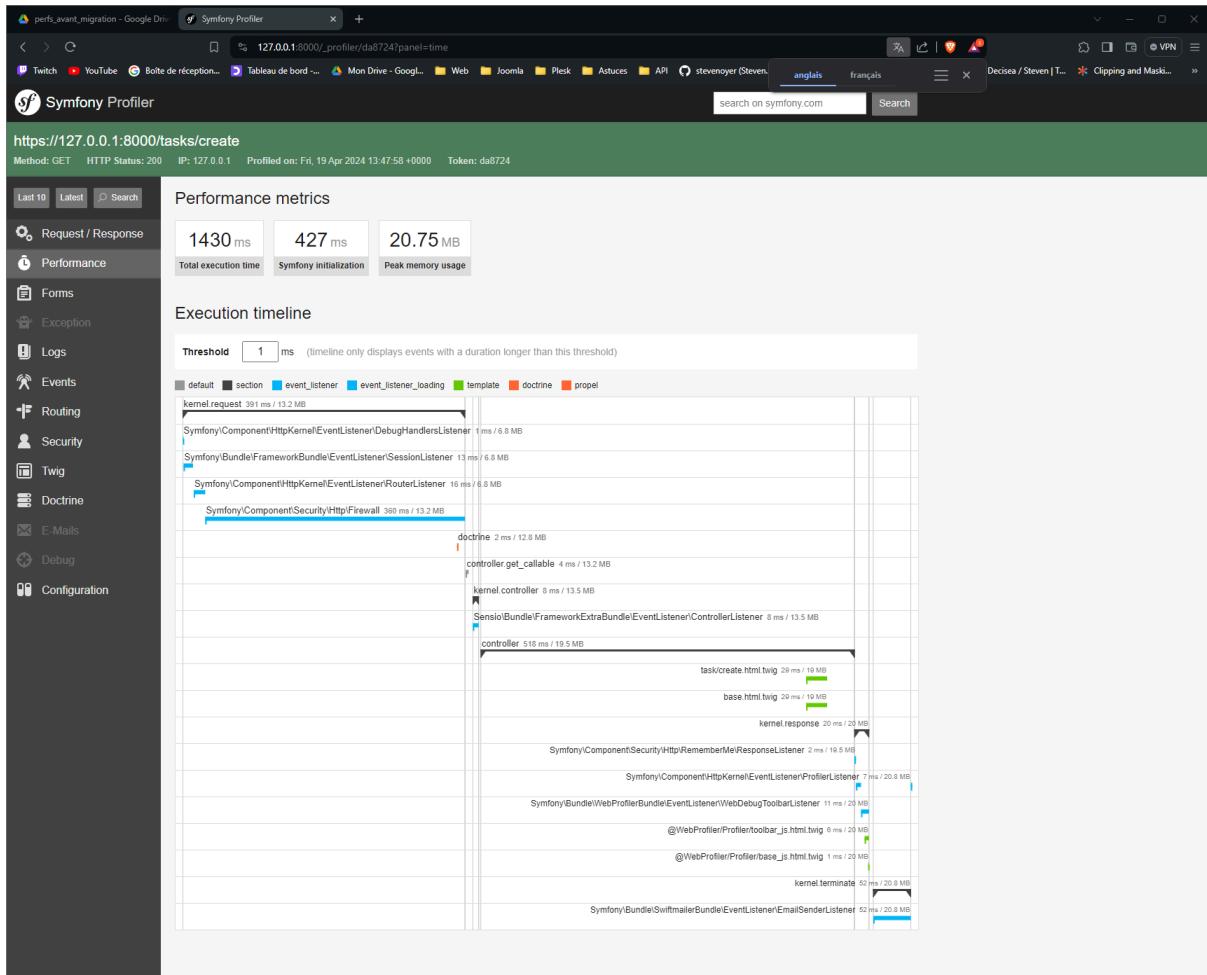
Home



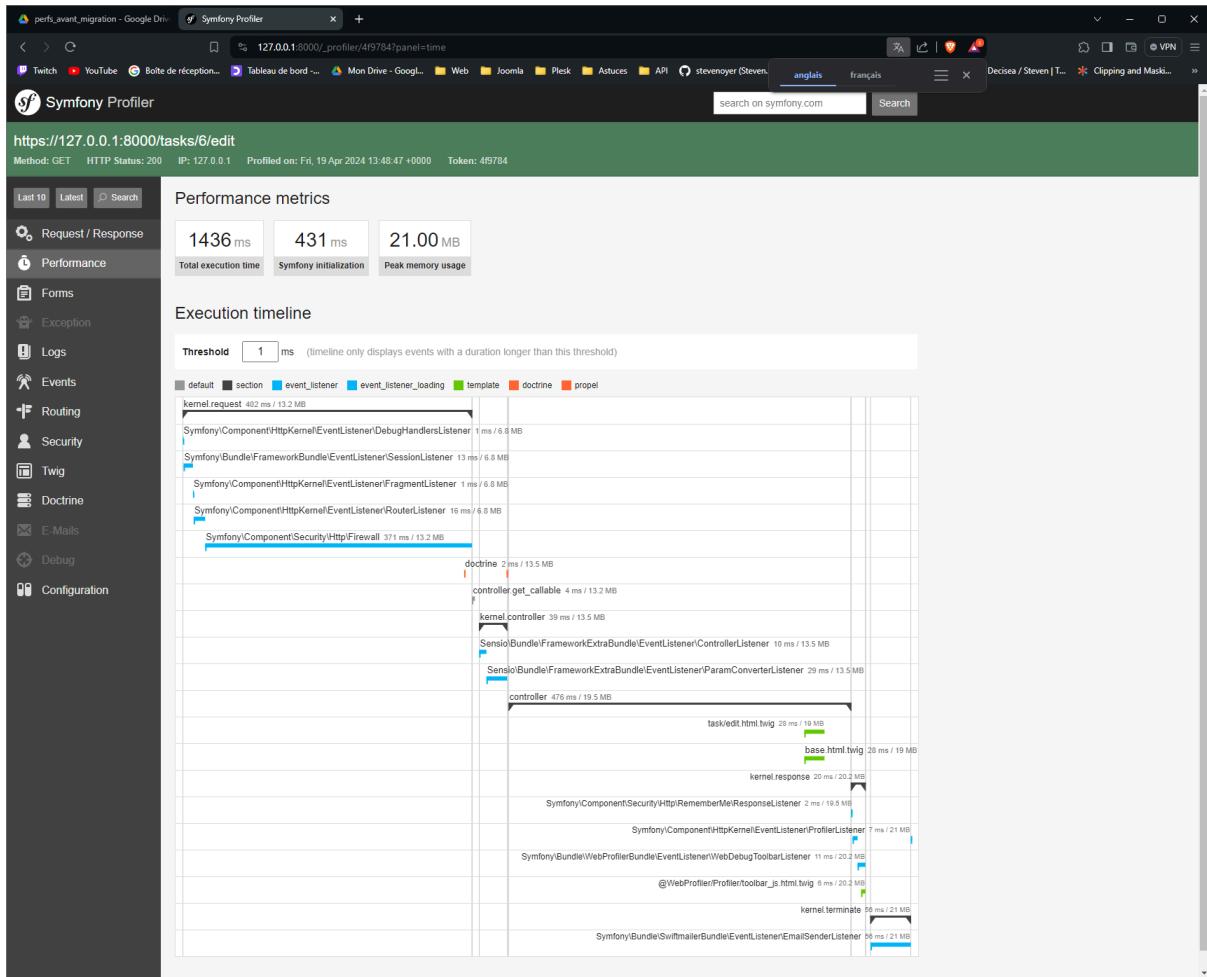
Tasks



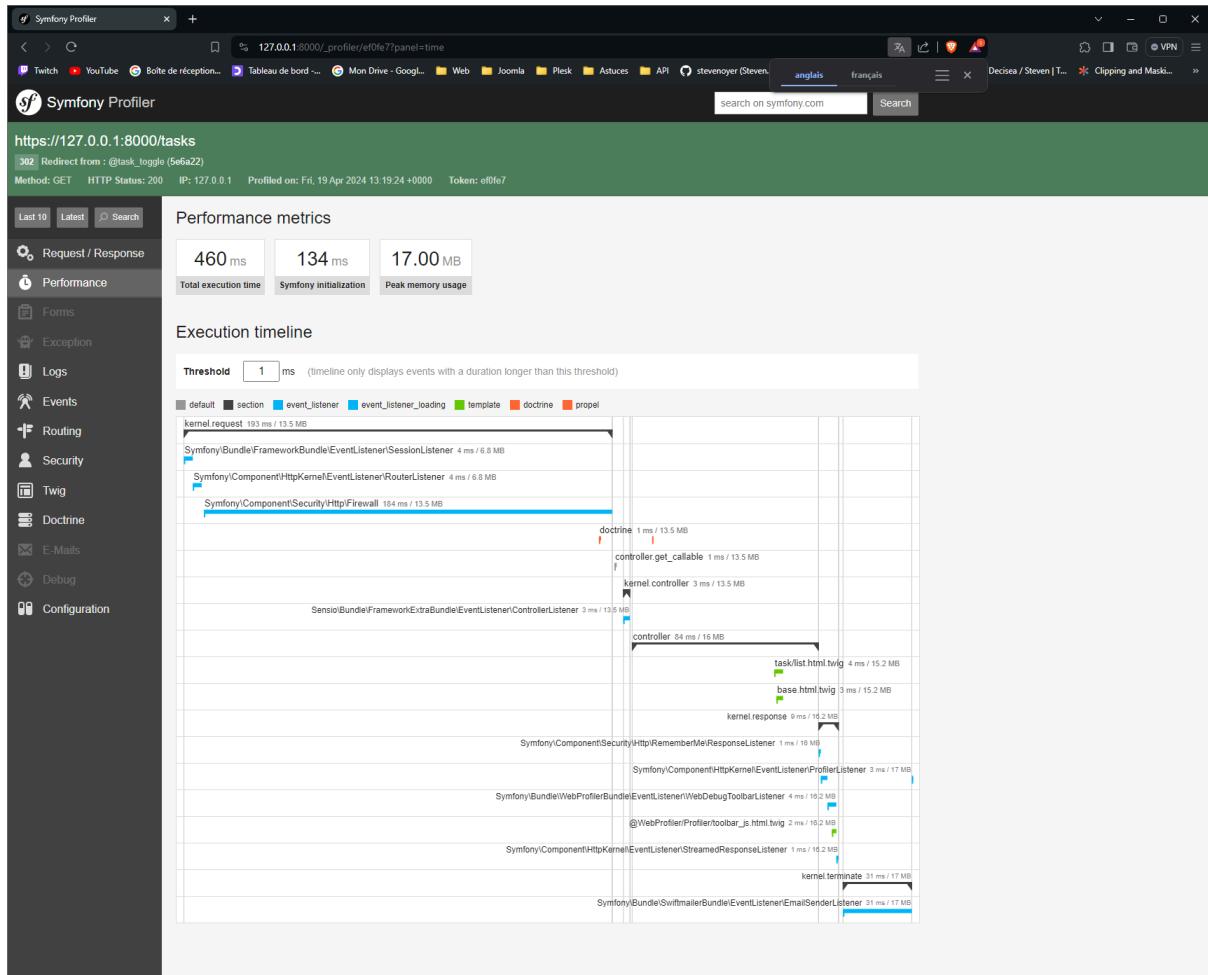
Task create



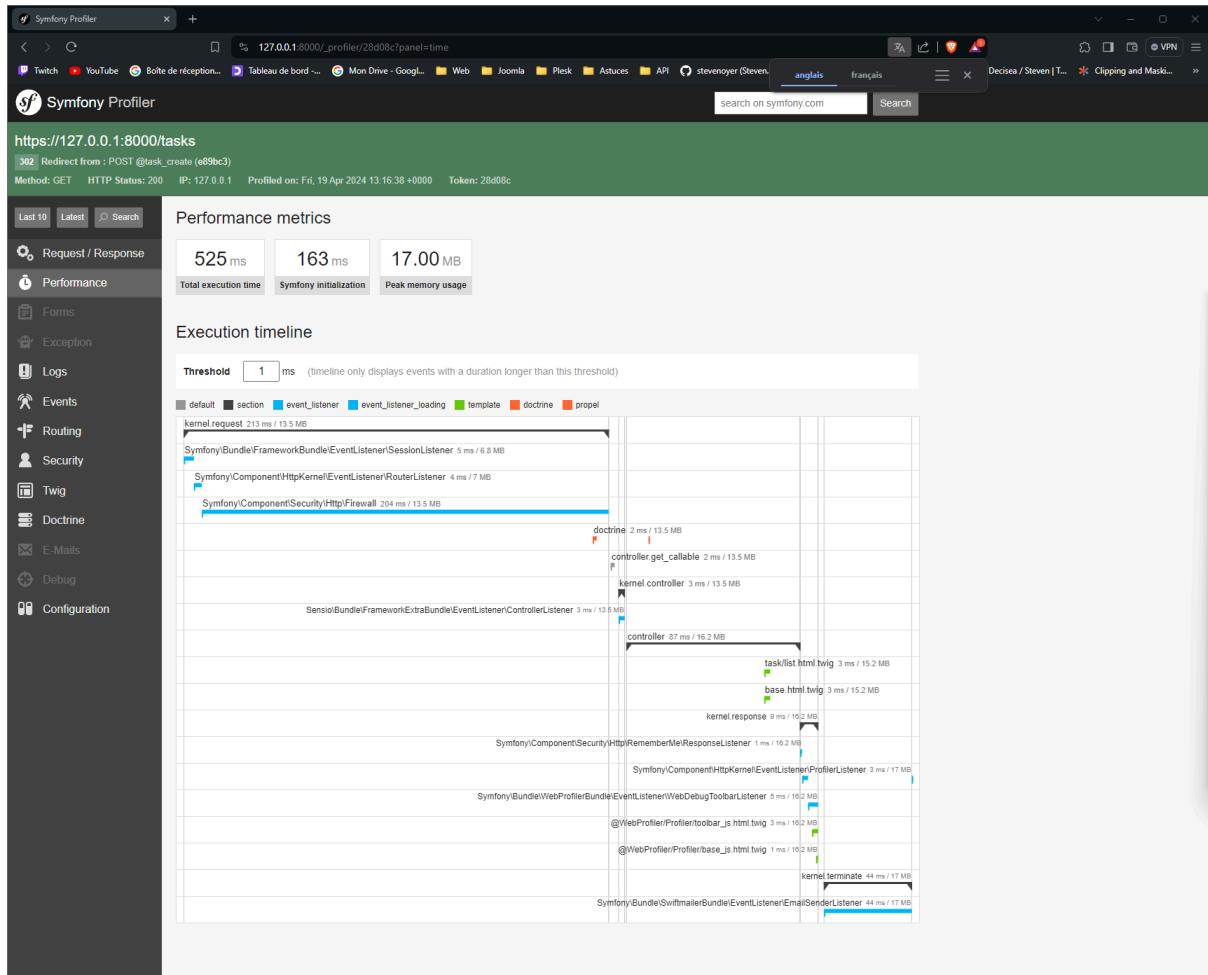
Task edit



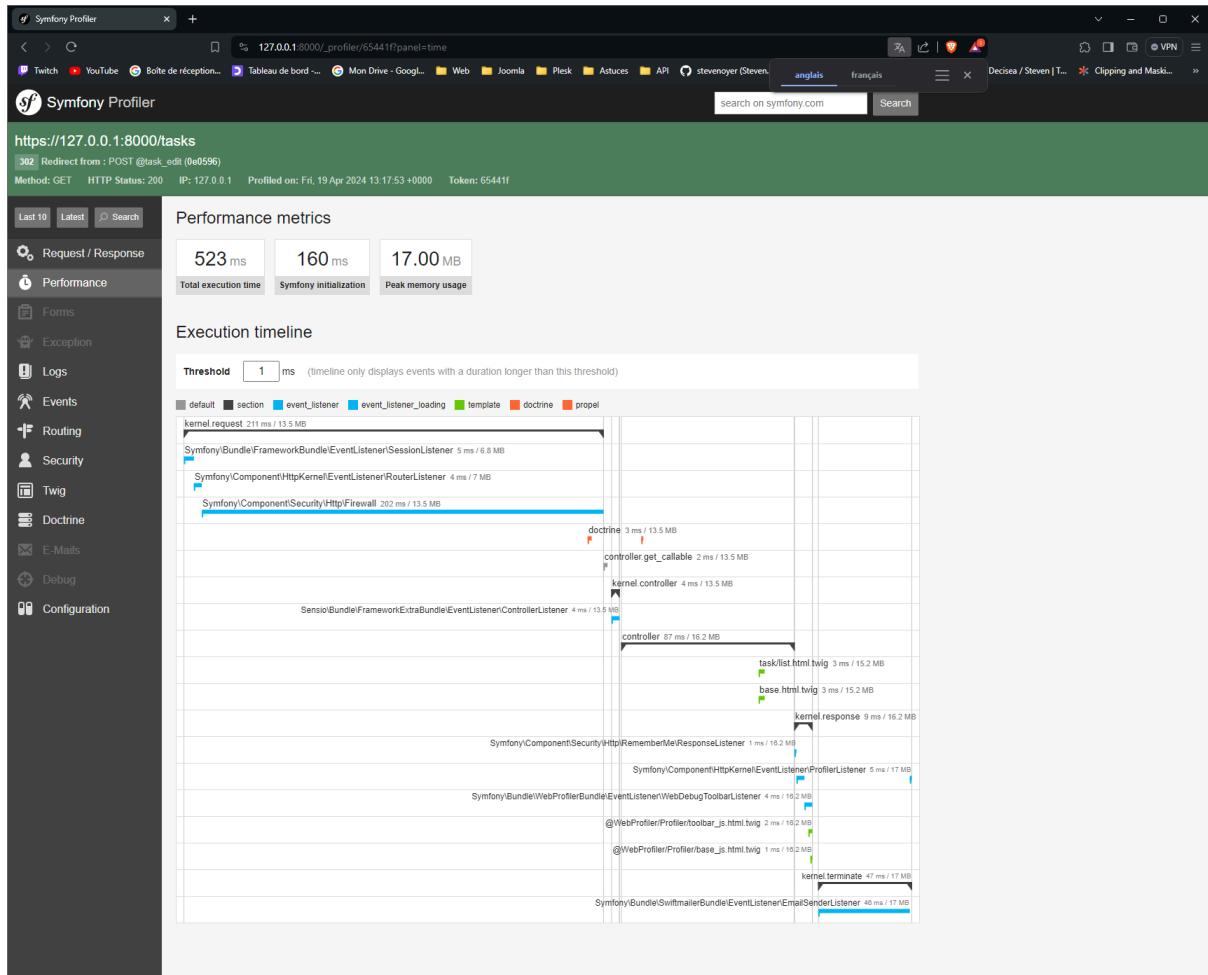
Tasks after change status



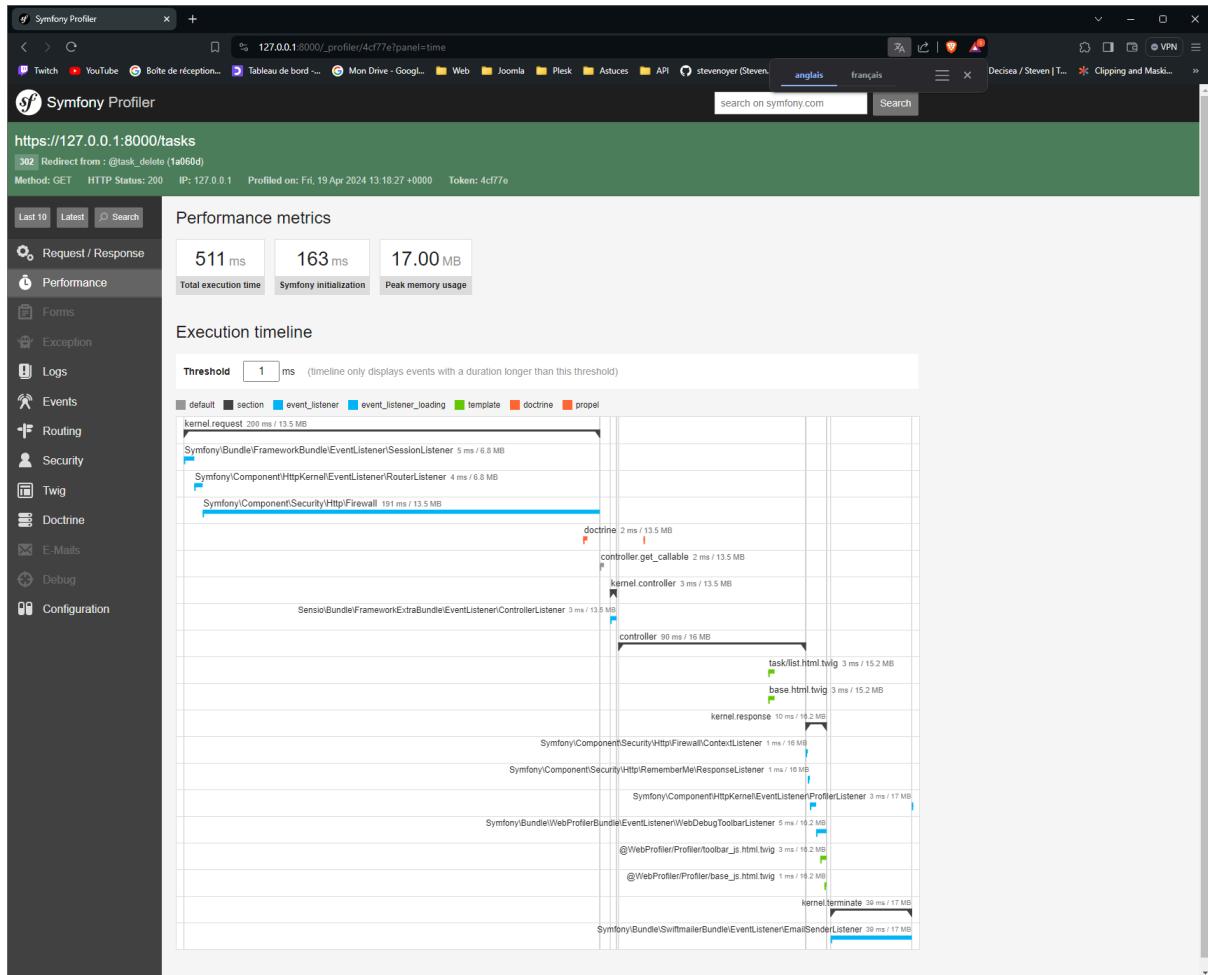
Tasks after create



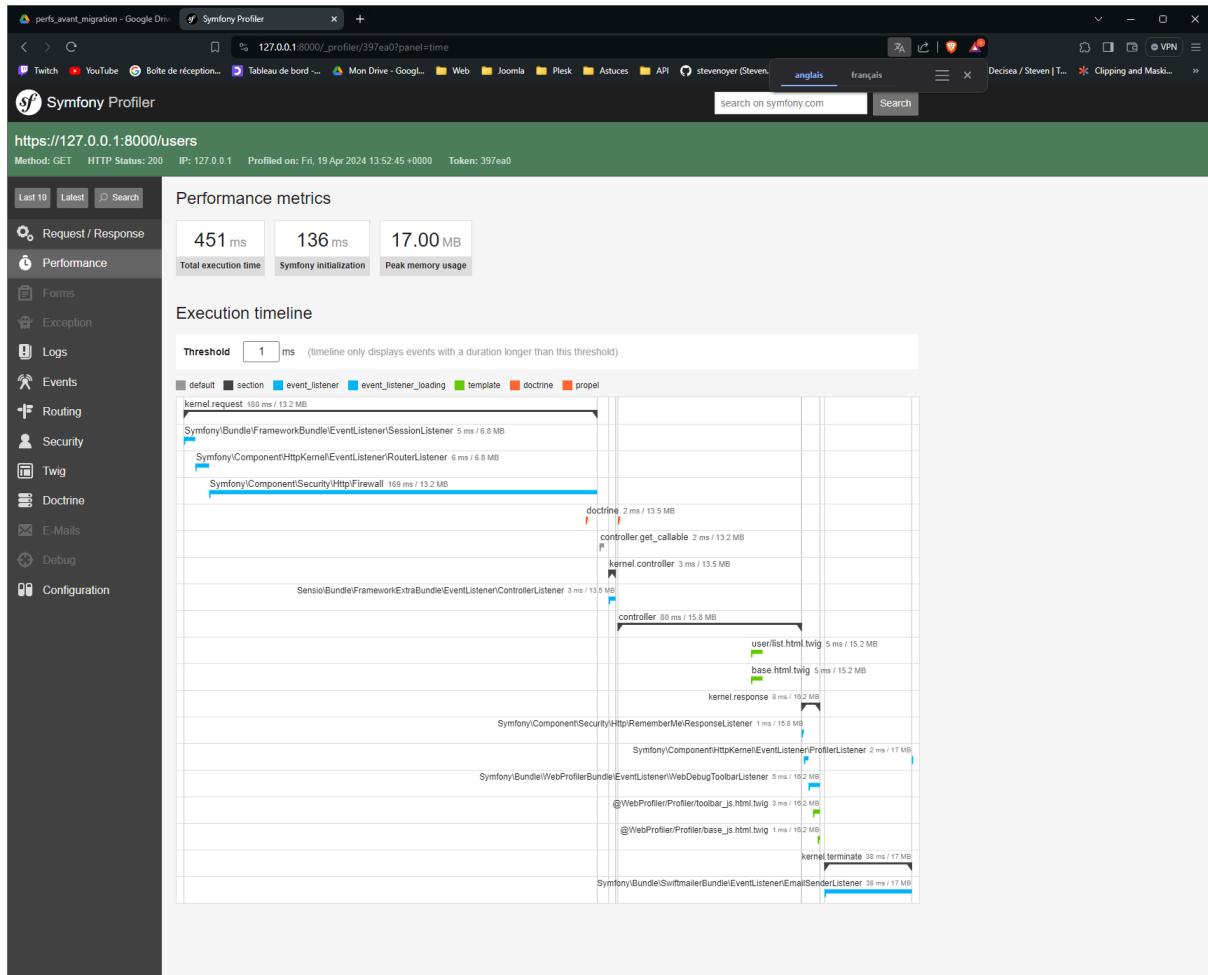
Tasks after edit



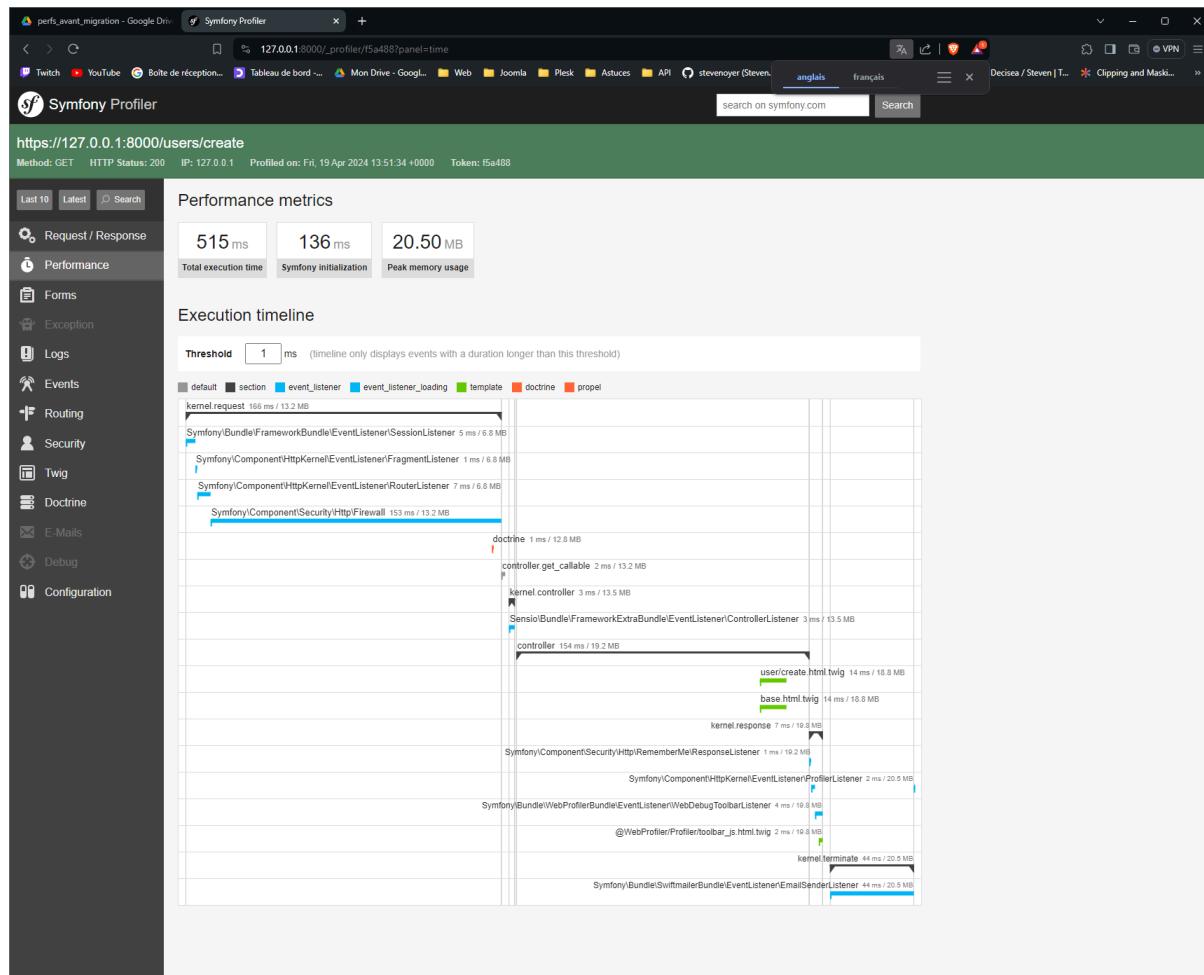
Tasks after delete



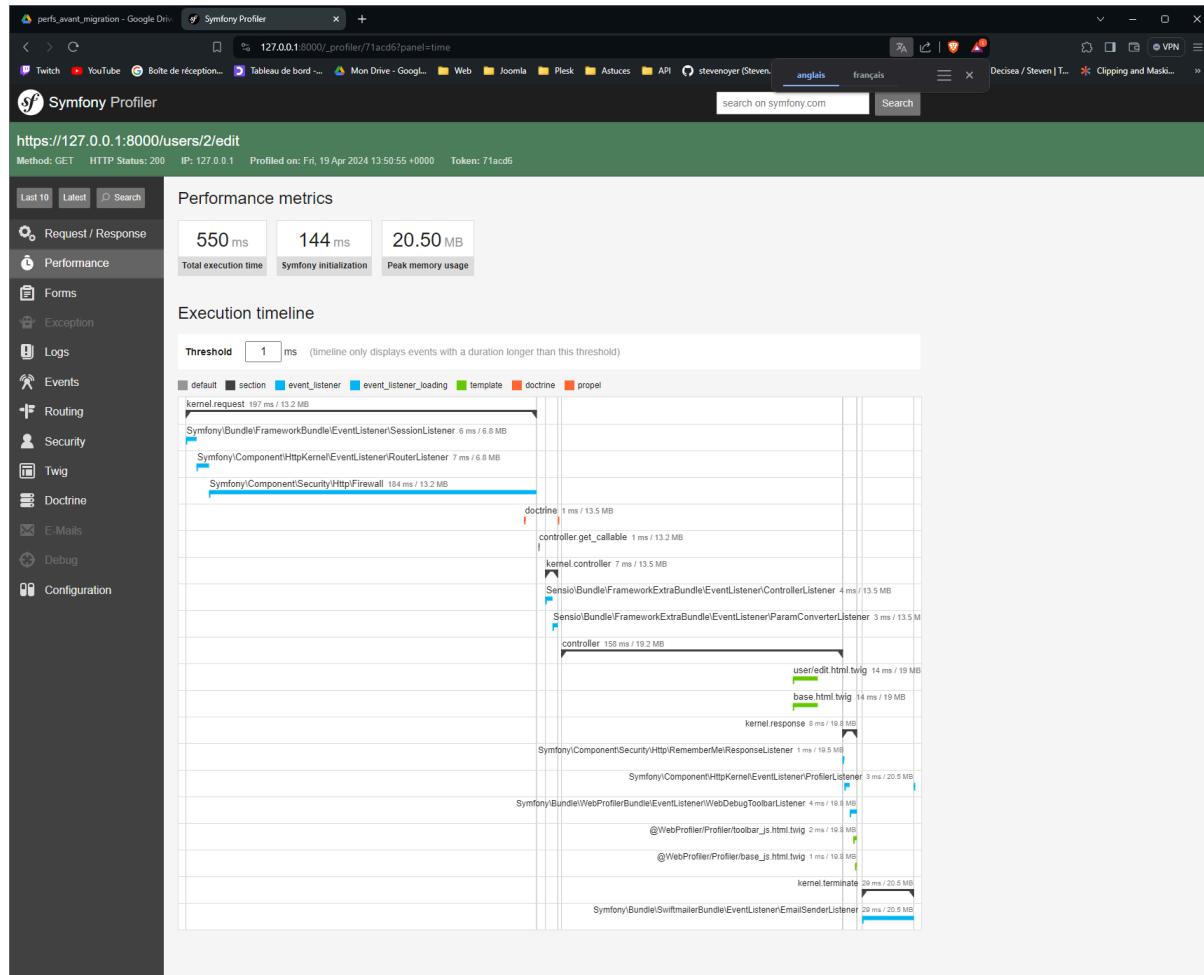
Users



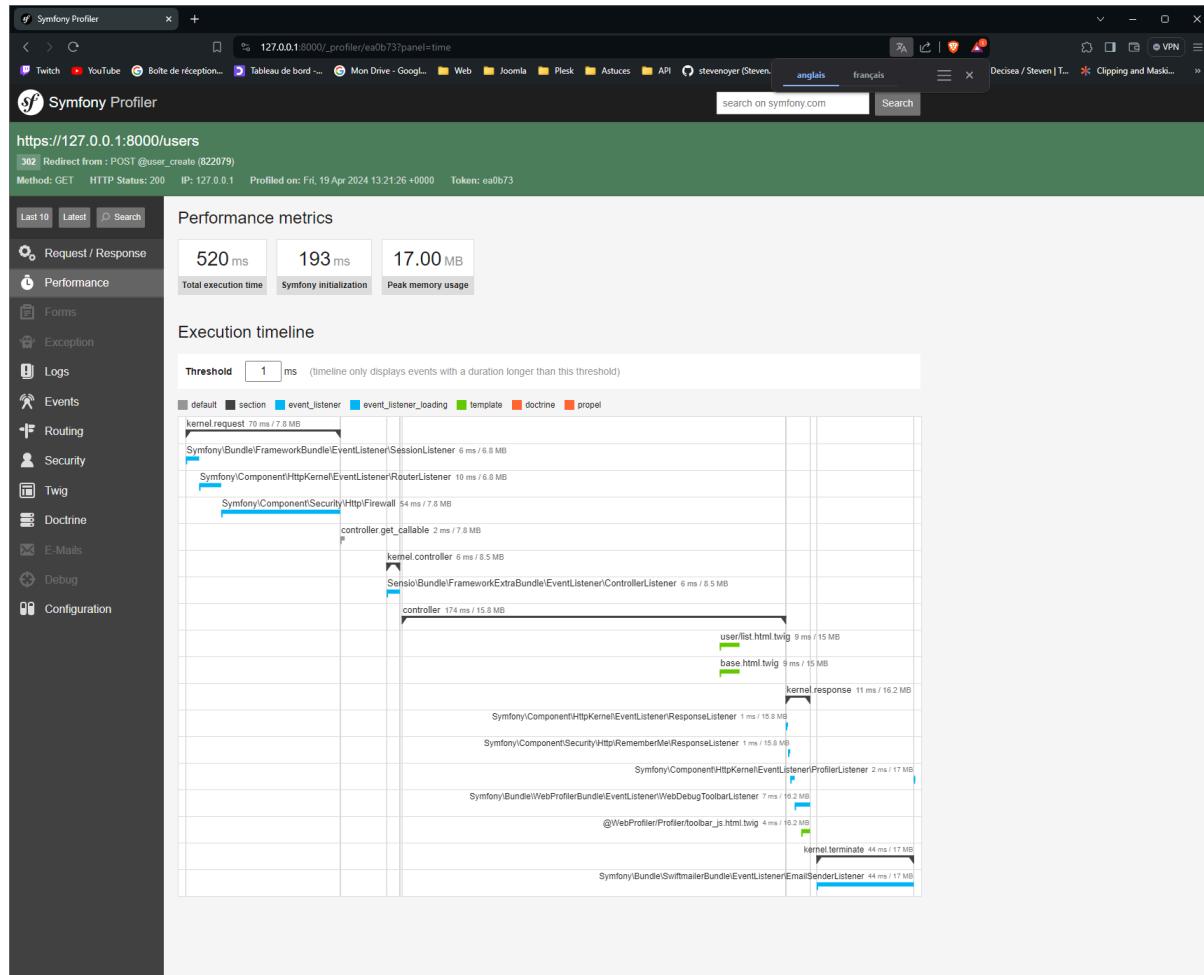
Create user



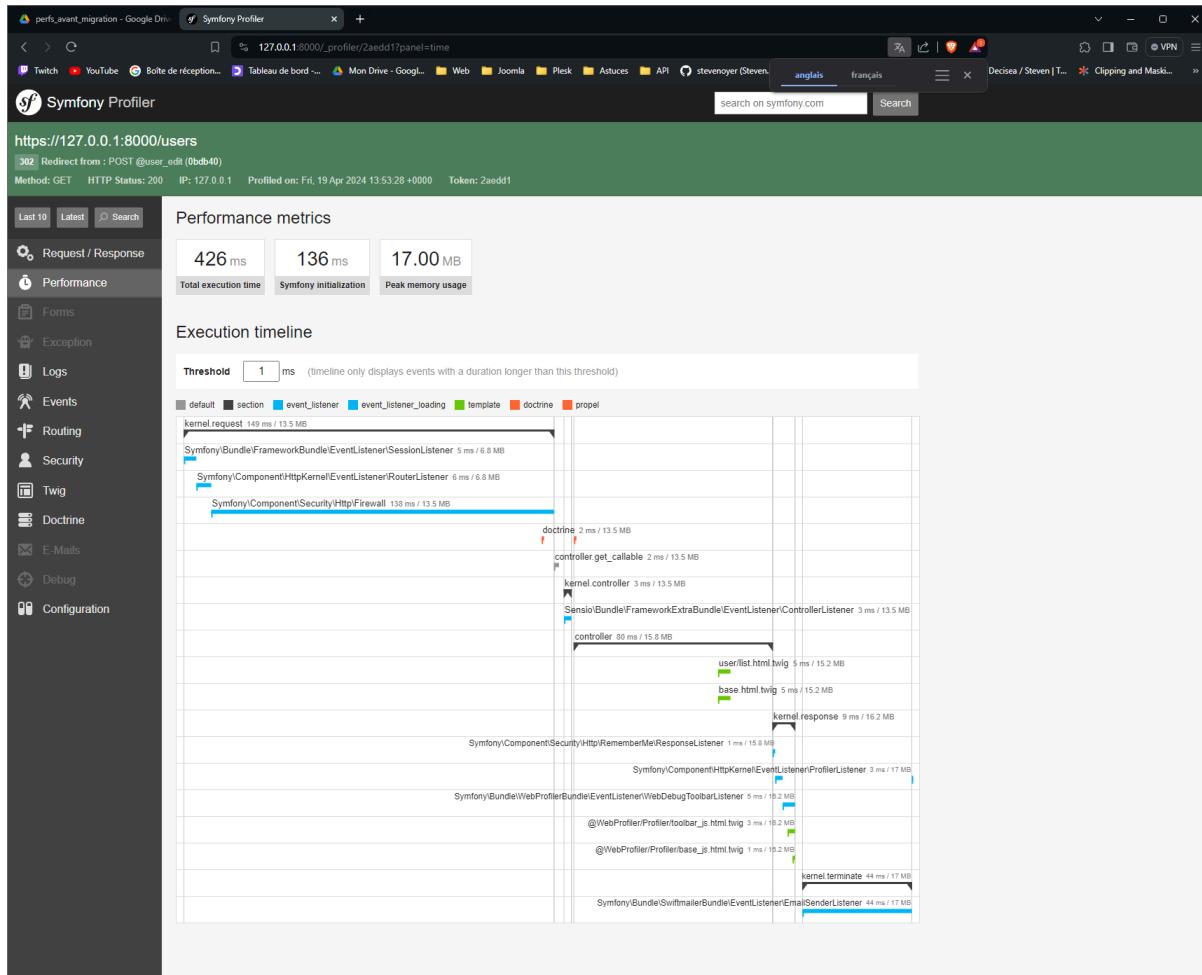
Edit user



Users after create

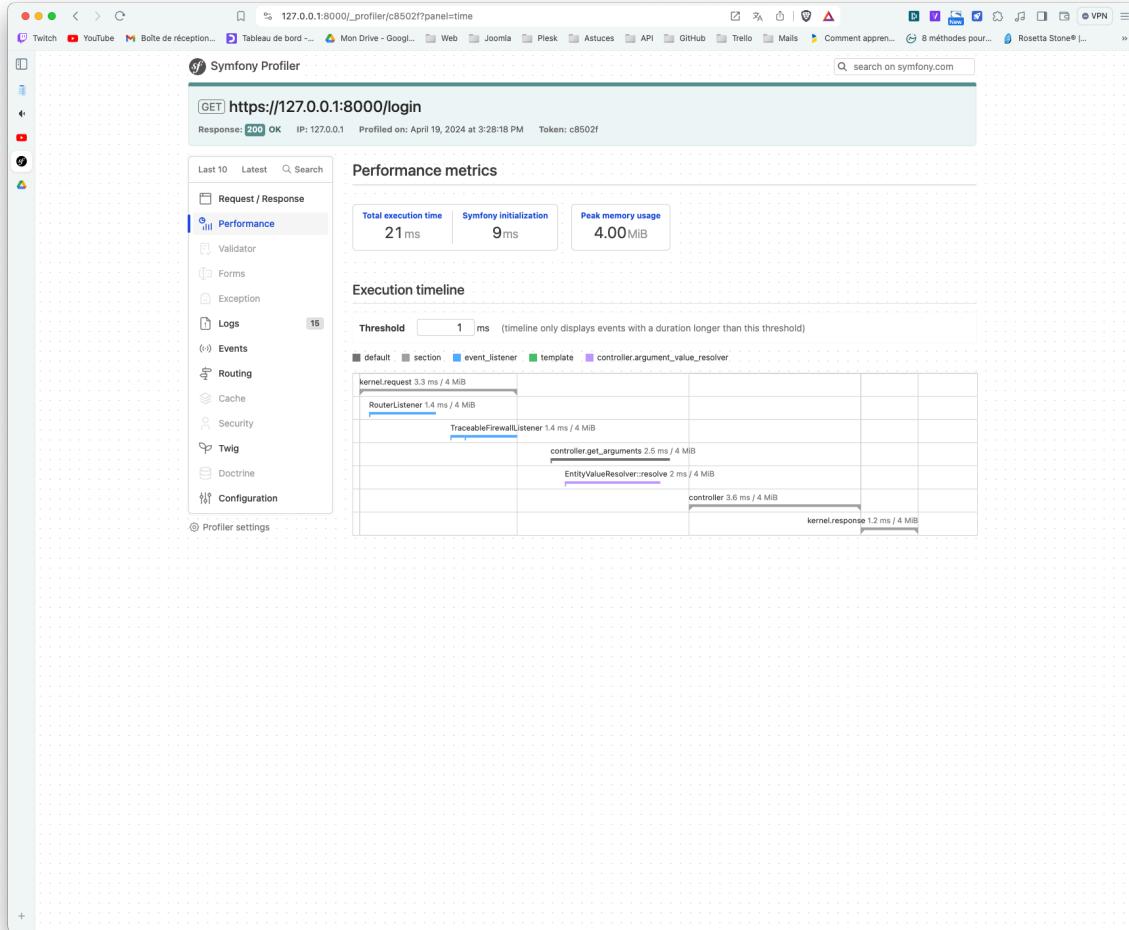


Users after edit



Symfony 6.4

Login



Login after disconnect

The screenshot shows the Symfony Profiler interface for a request to `https://127.0.0.1:8000/login`. The request was successful (200 OK) and was profiled on April 19, 2024 at 3:30:15 PM with a token of 3972ad.

Performance metrics:

- Total execution time: 8ms
- Symfony initialization: 2ms
- Peak memory usage: 6.00 MIB

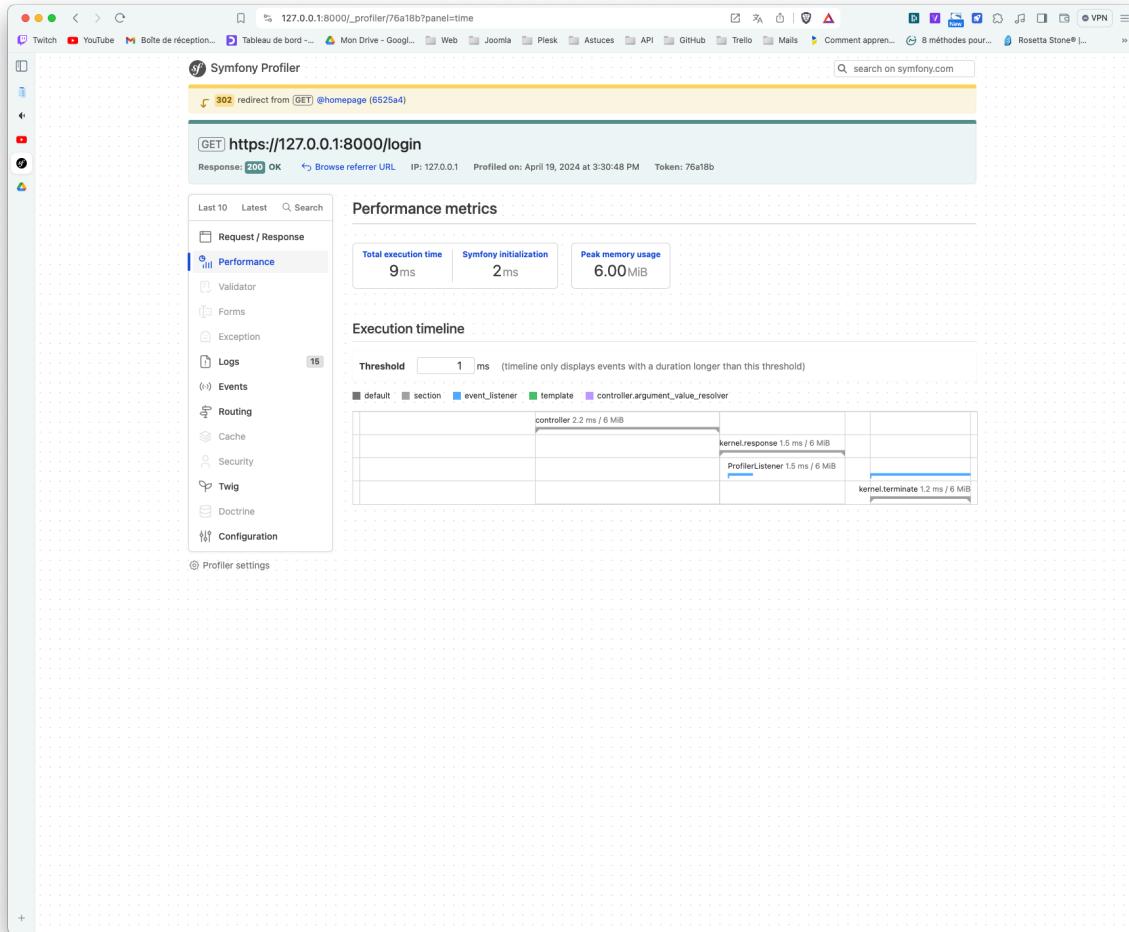
Execution timeline:

Threshold: 1 ms (timeline only displays events with a duration longer than this threshold)

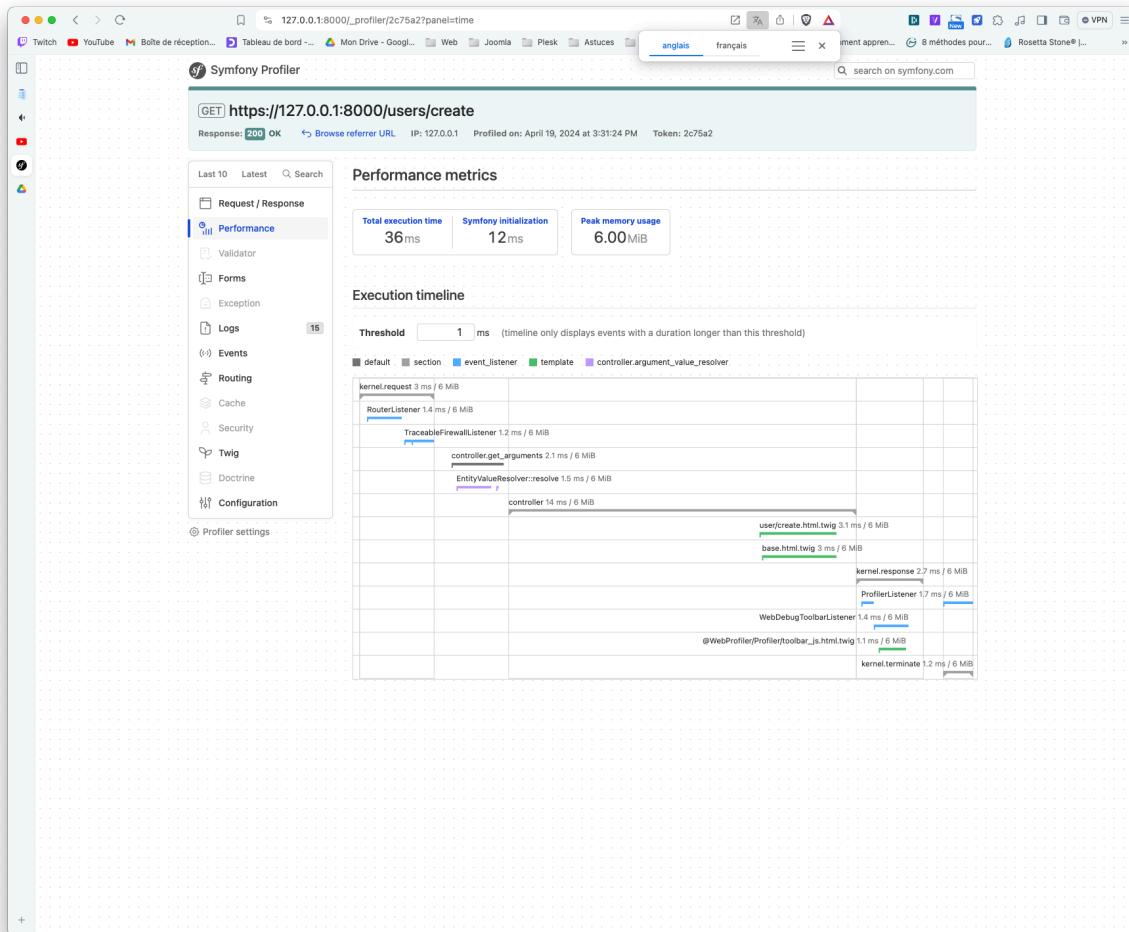
Legend: default, section, event_listener, template, controller.argument_value_resolver

Event	Duration	Memory Usage
controller.get_arguments	1 ms	6 MIB
controller	1.9 ms	6 MIB
kernel.response	1.3 ms	6 MIB

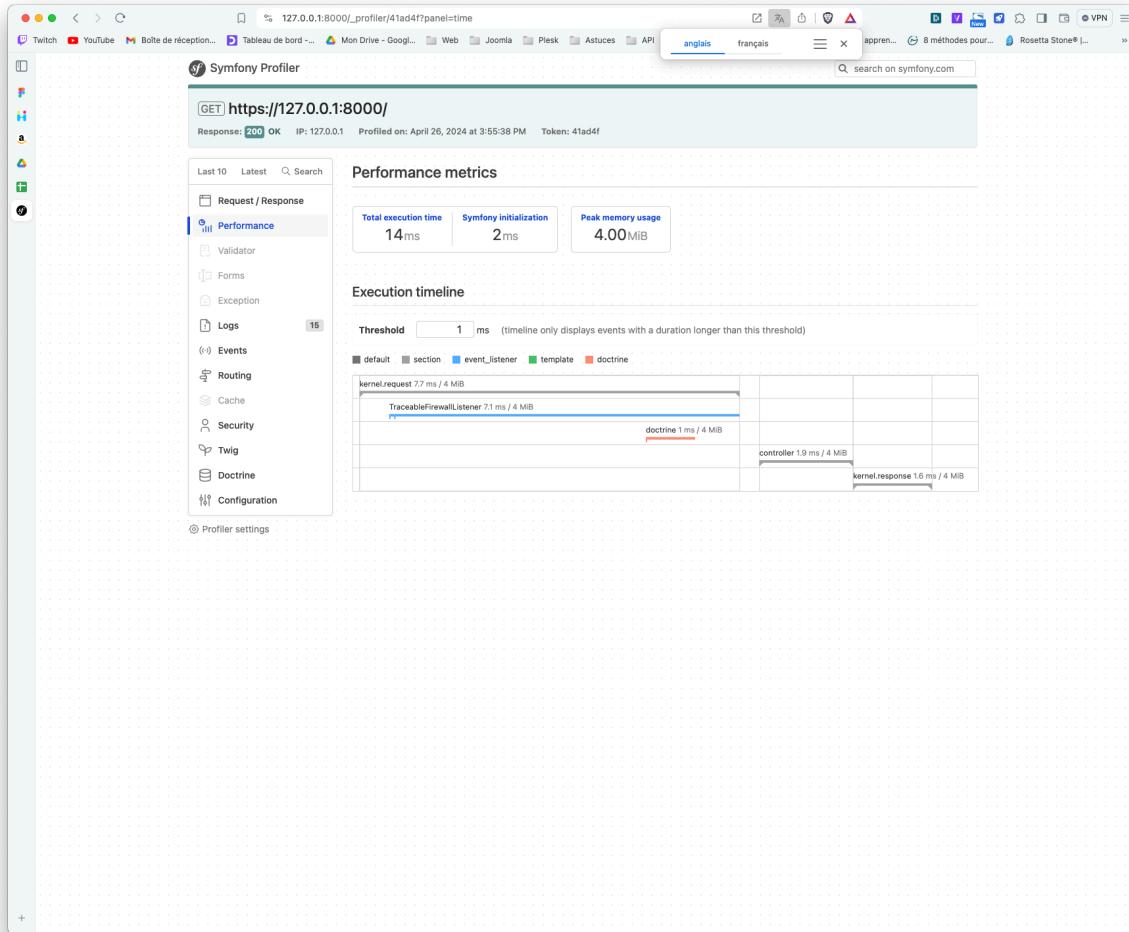
Login after register



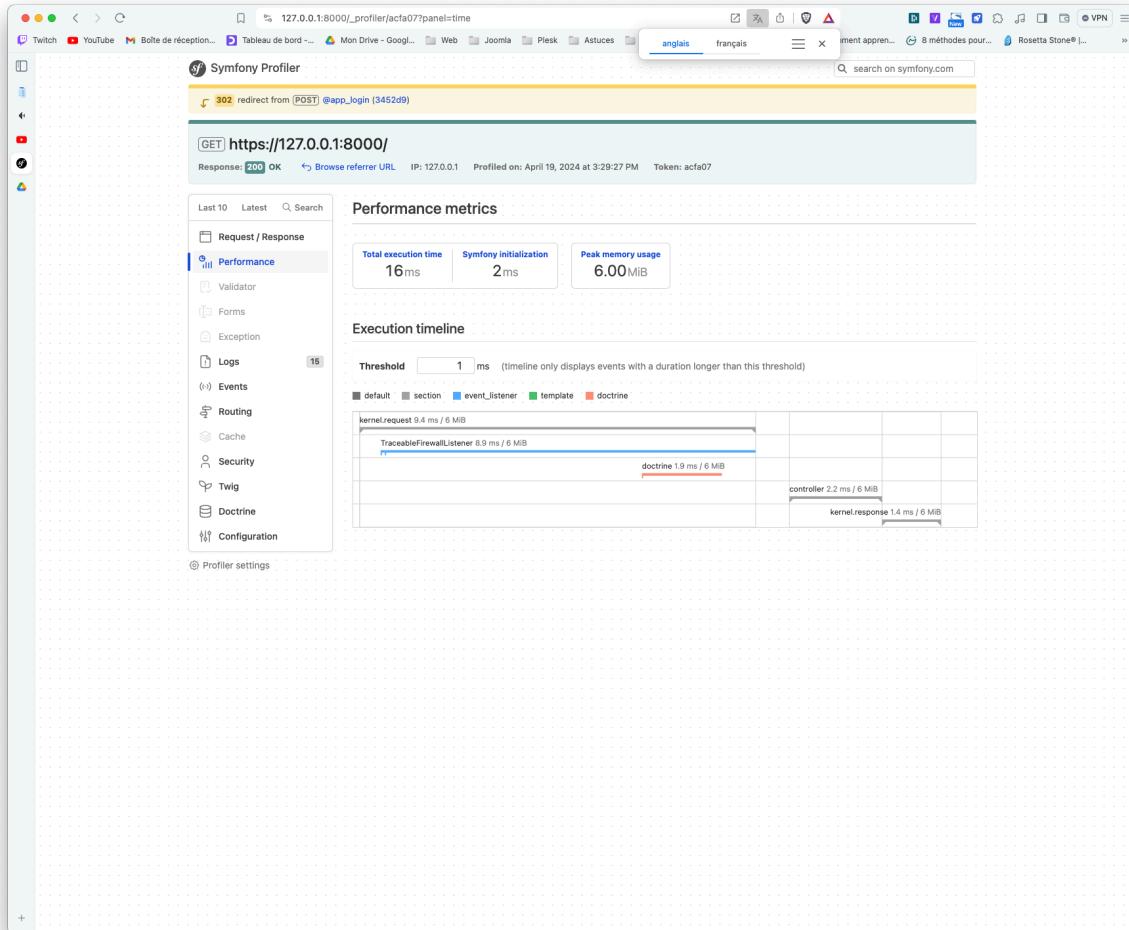
Register



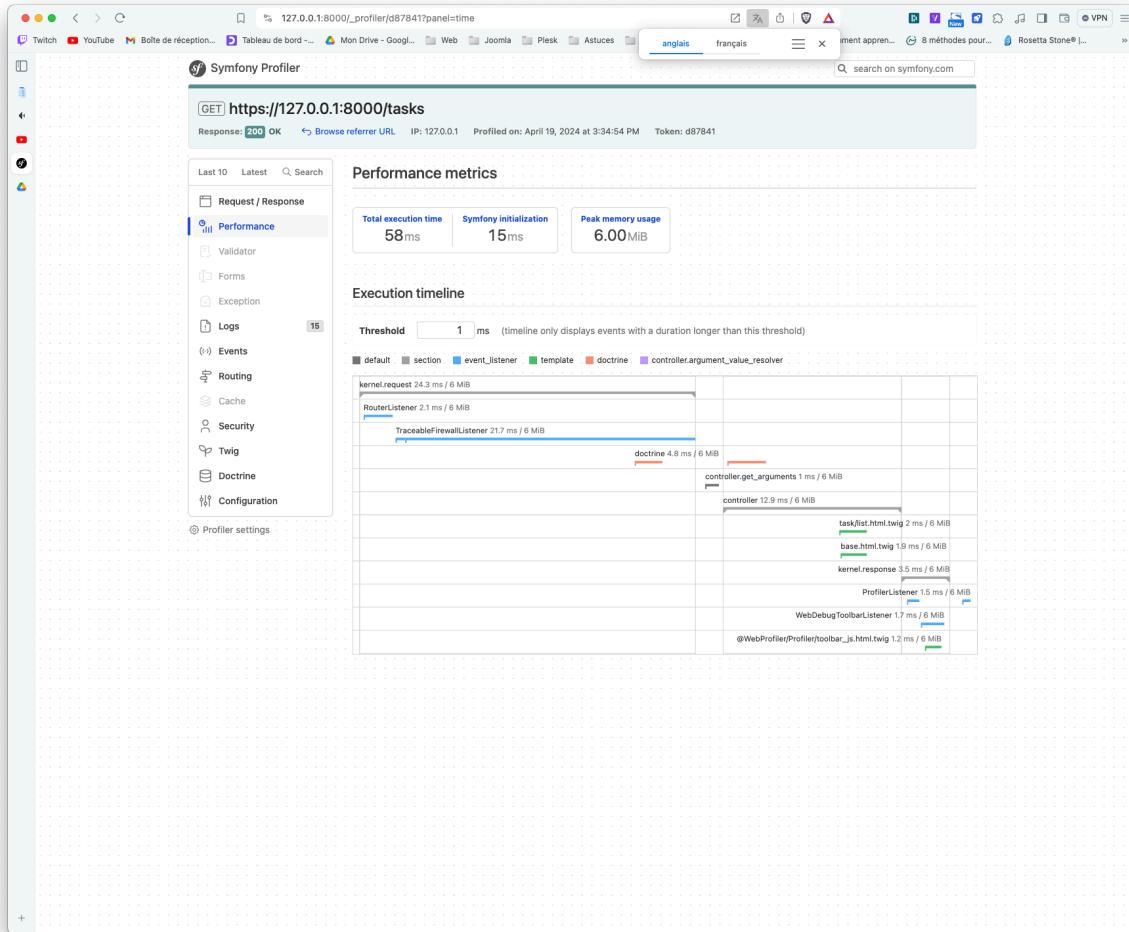
Home



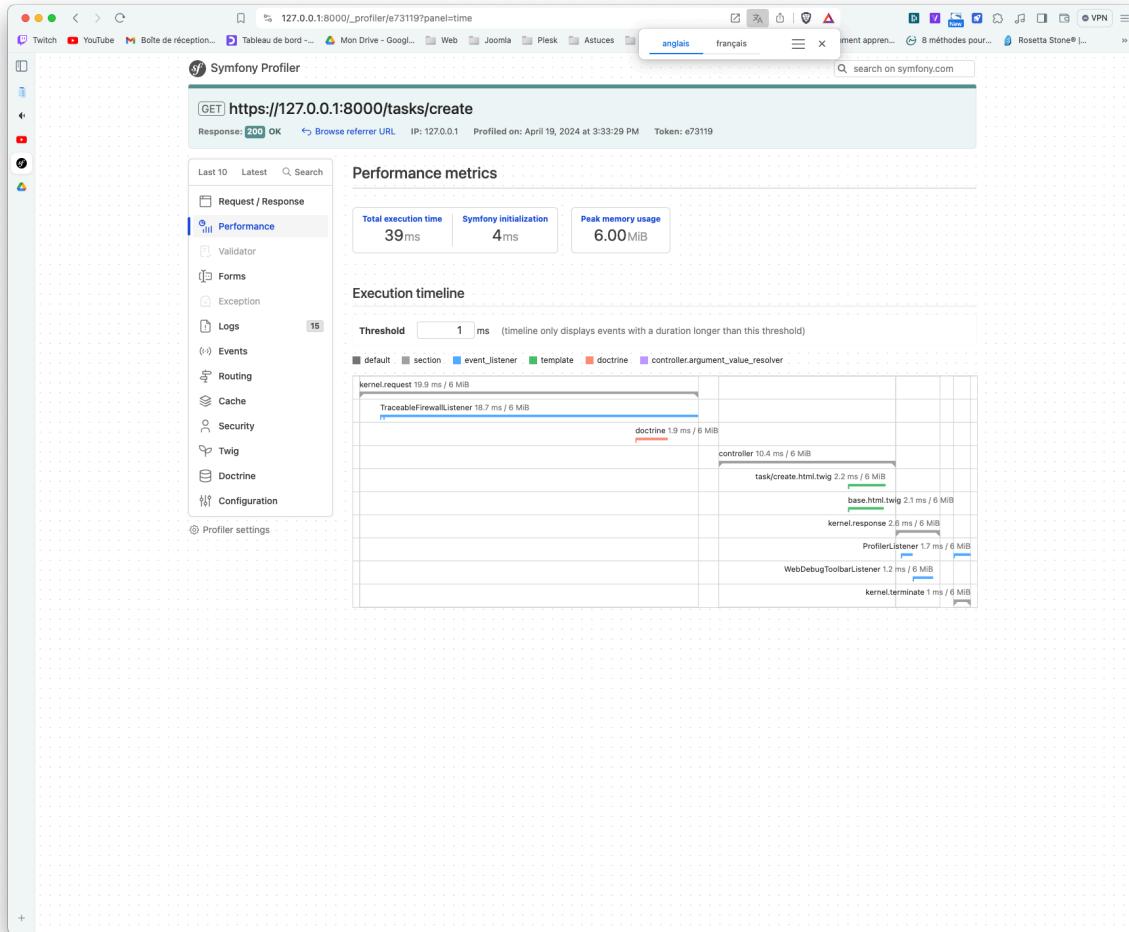
Home after login



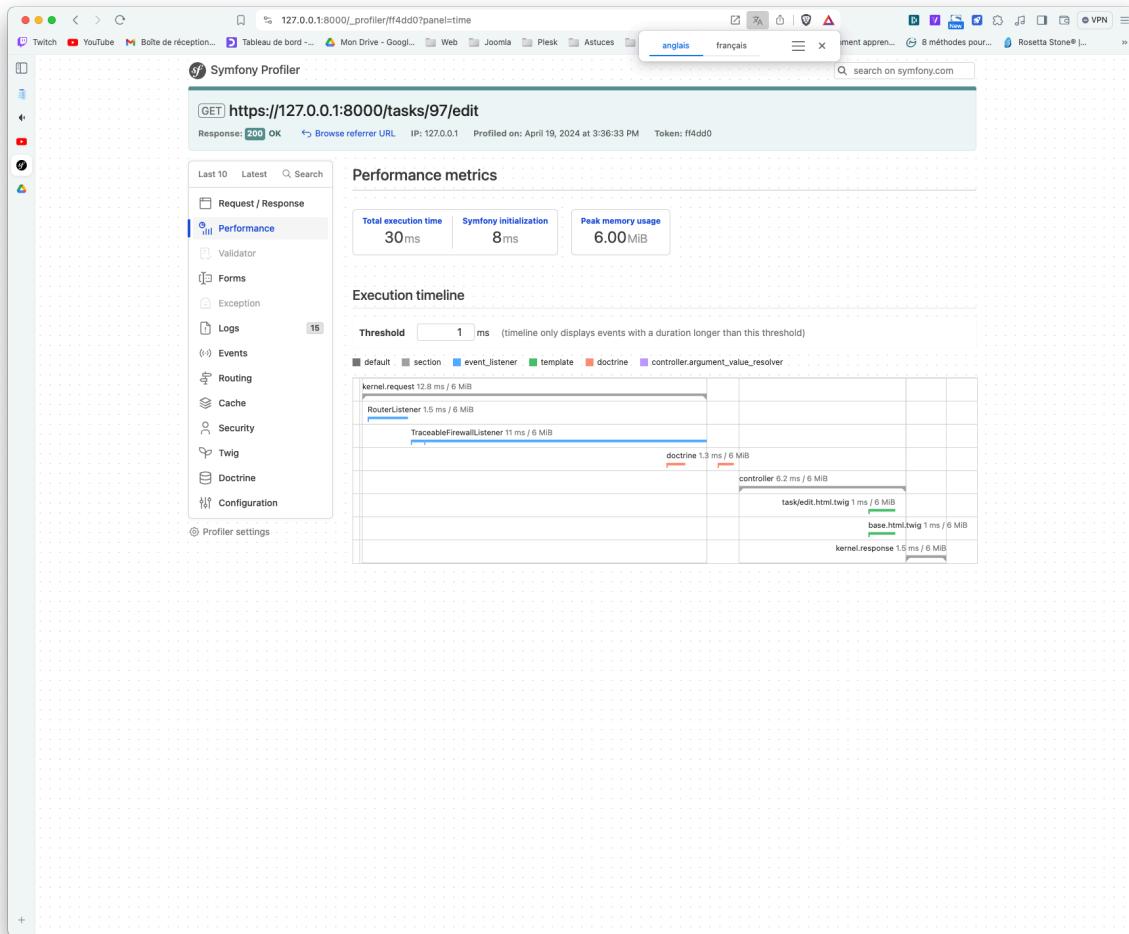
Tasks



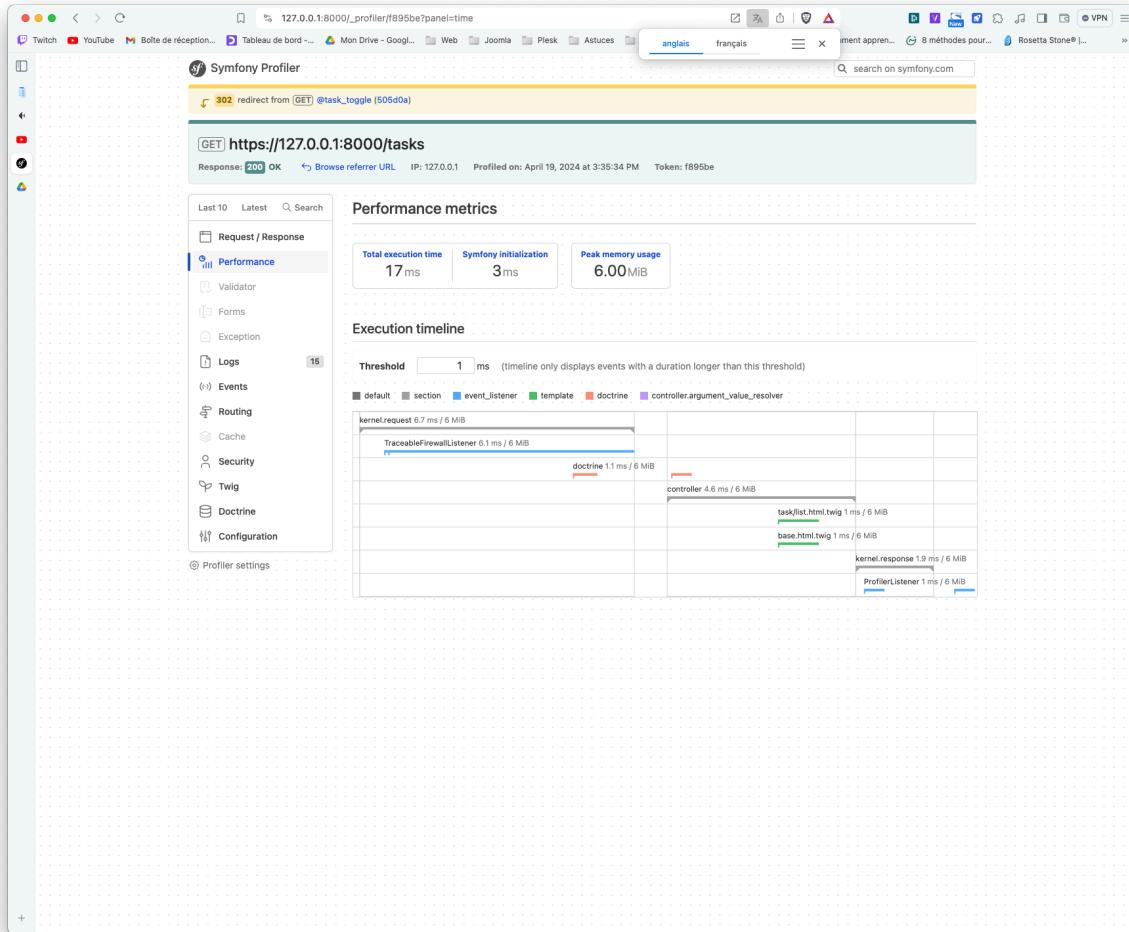
Task create



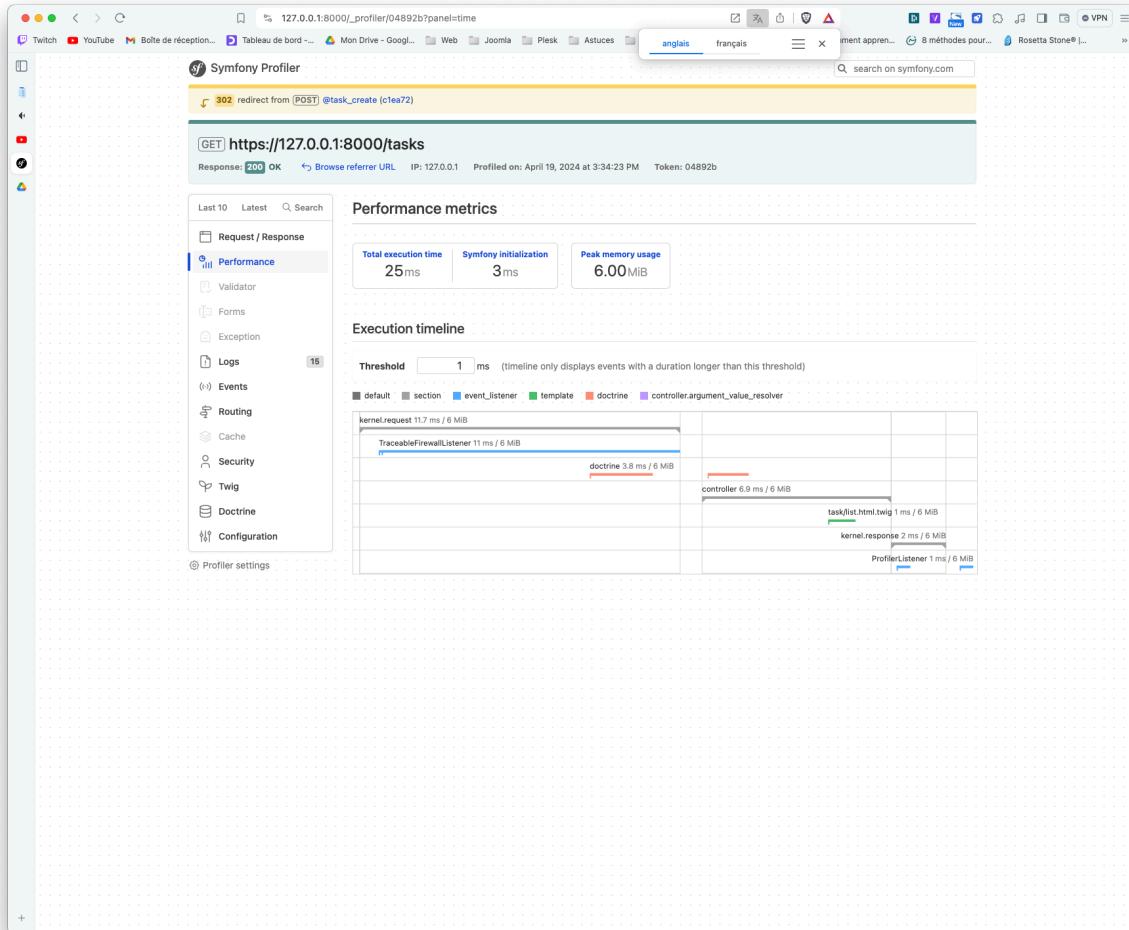
Task edit



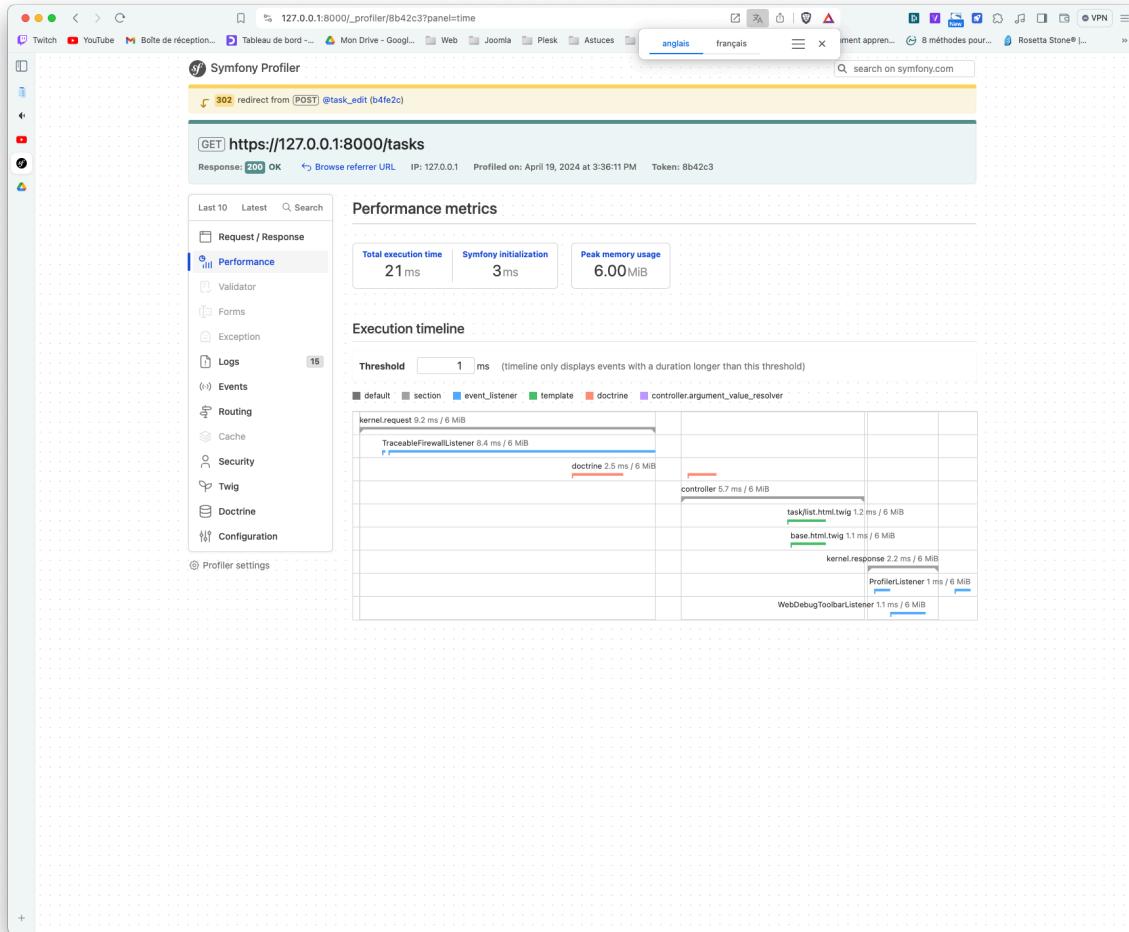
Tasks after change status



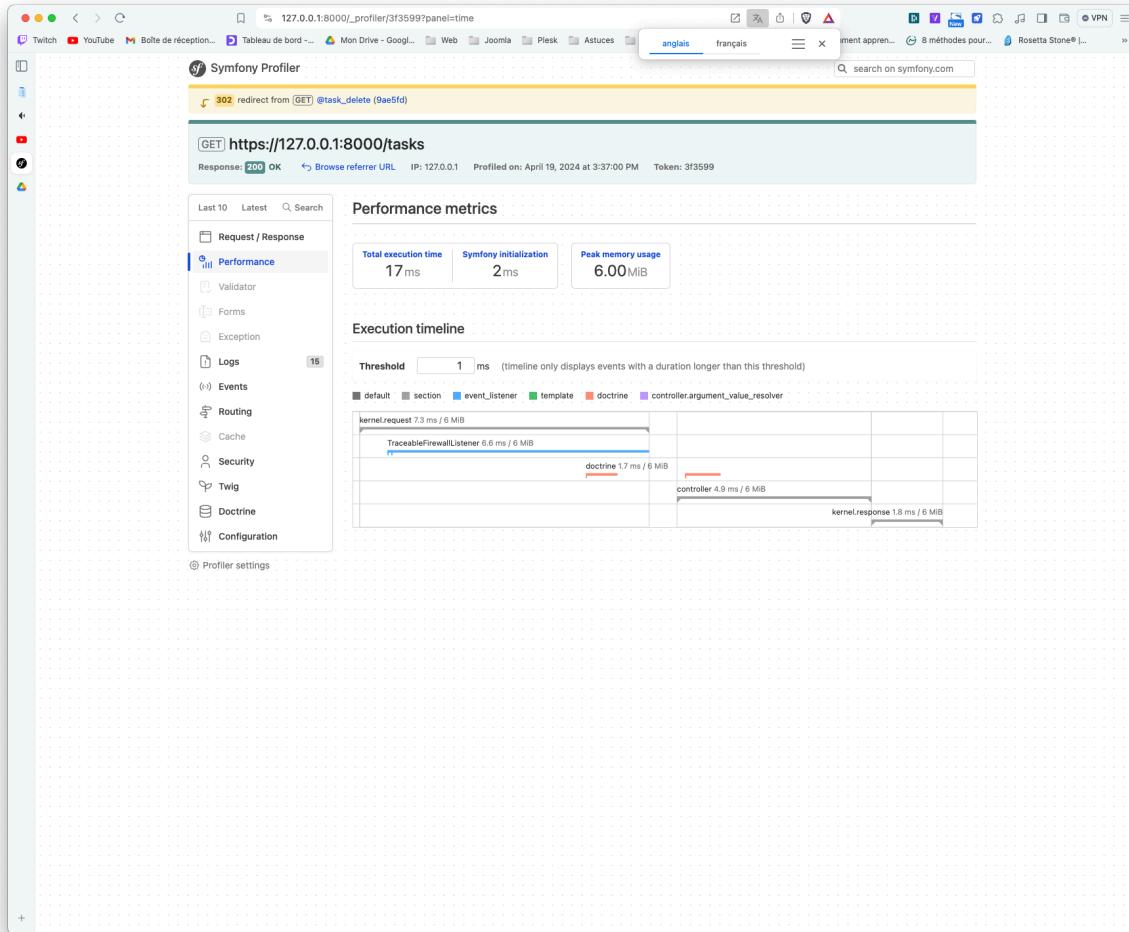
Tasks after create



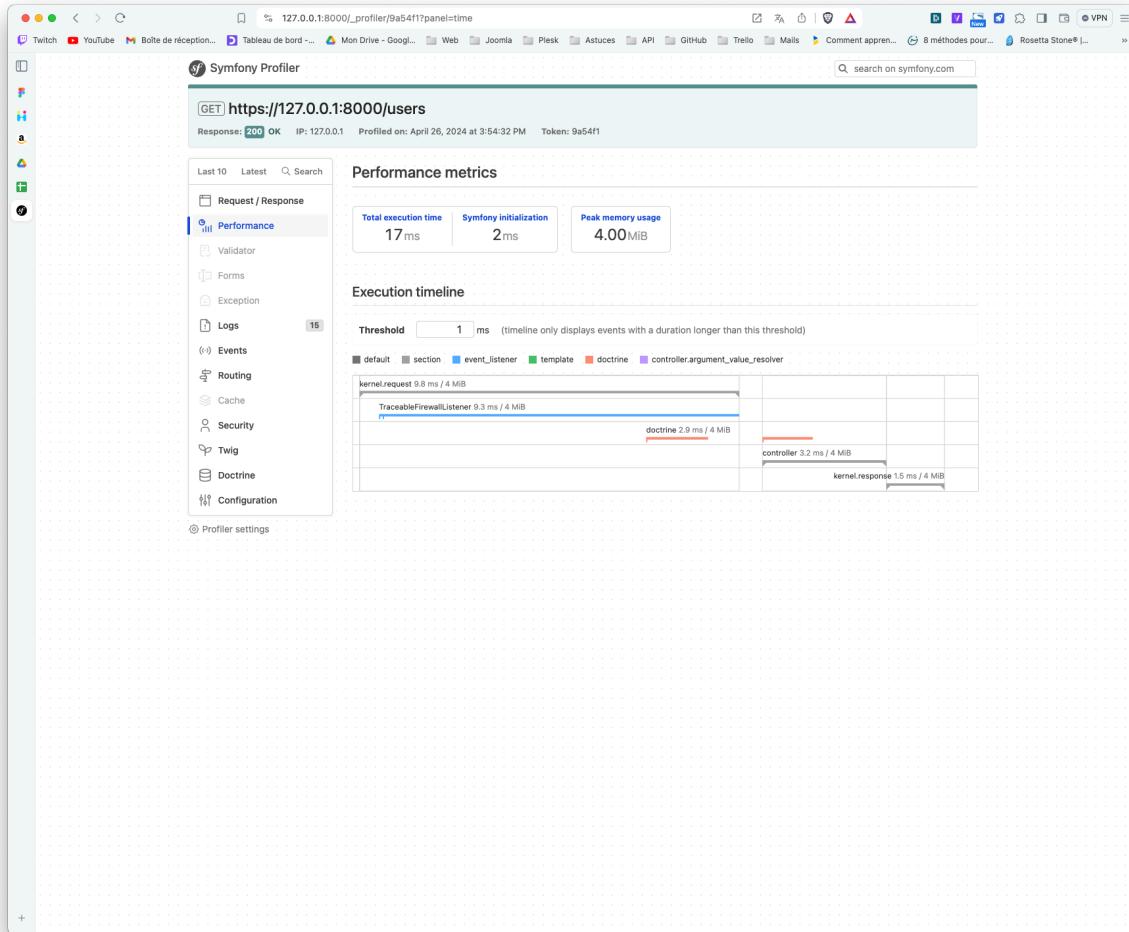
Tasks after edit



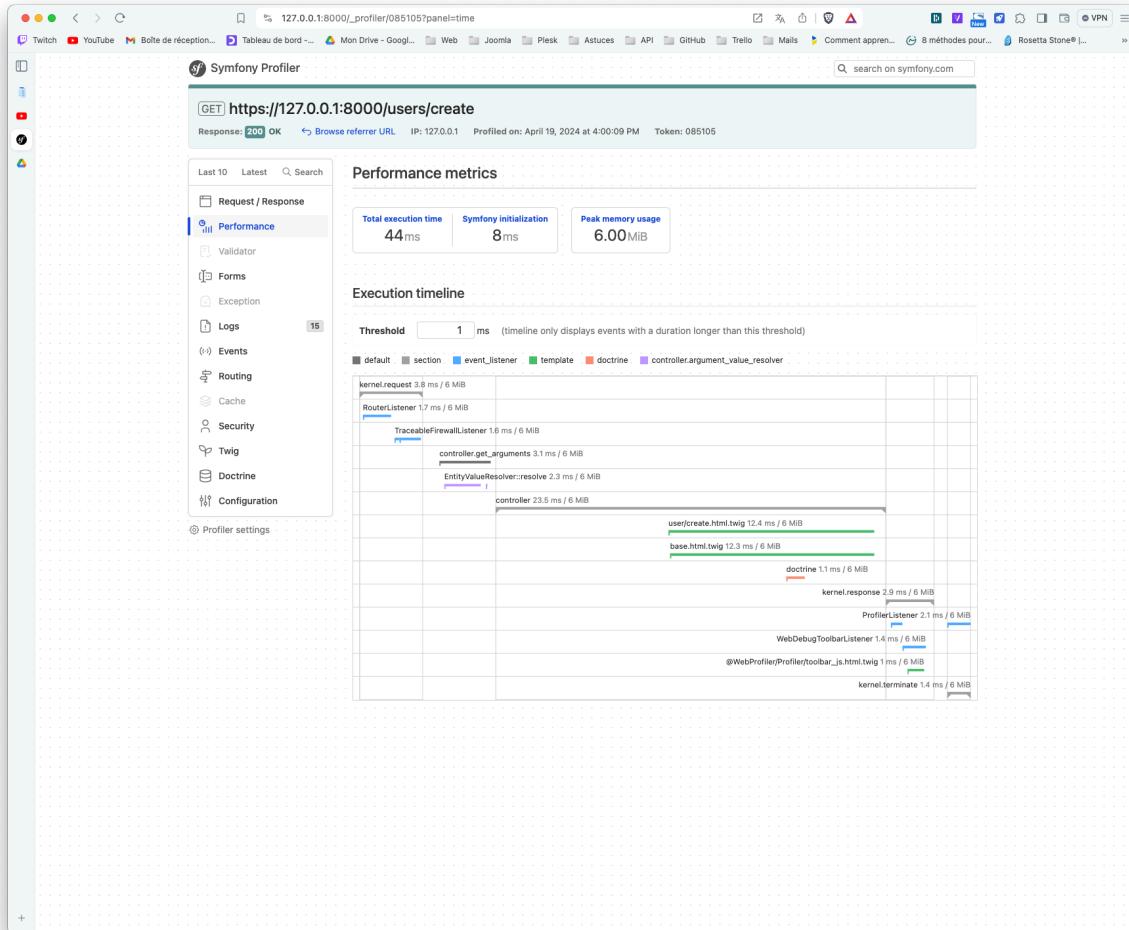
Tasks after delete



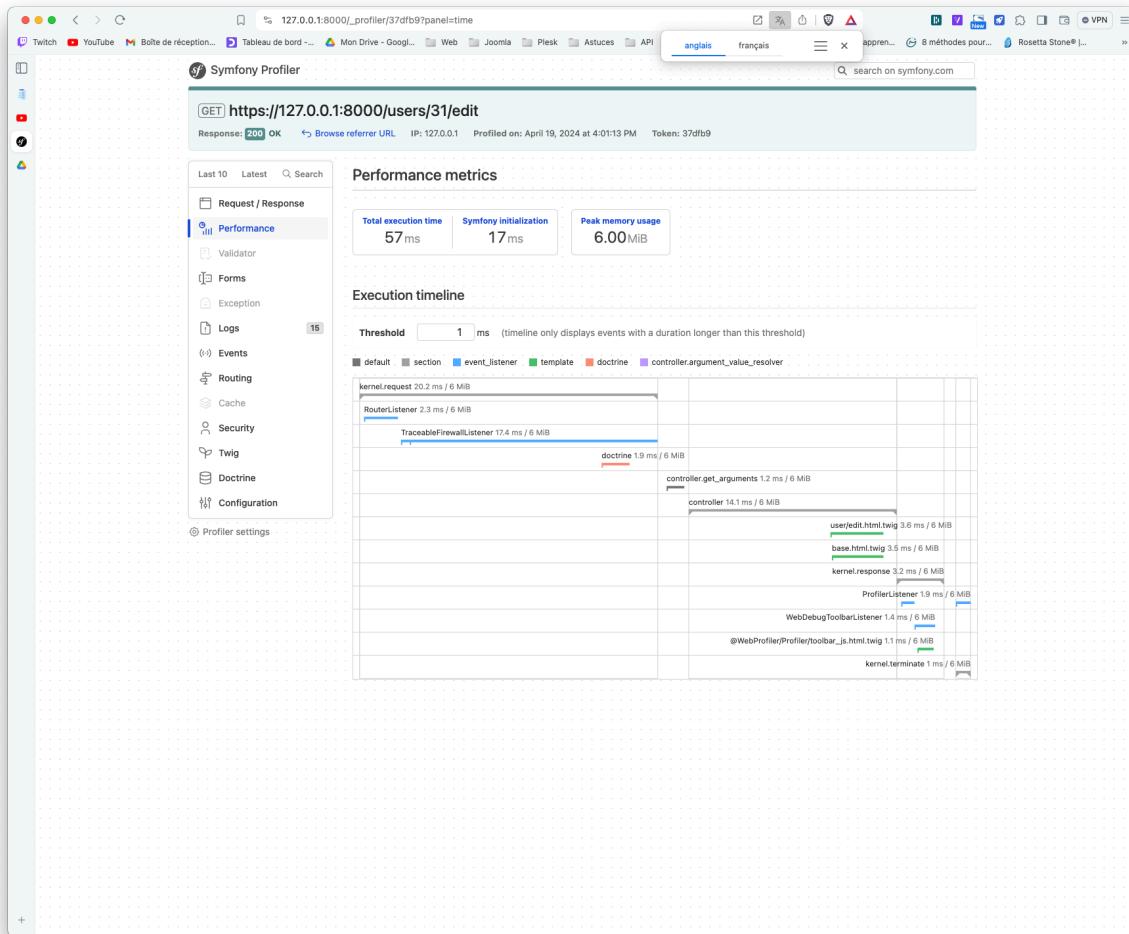
Users



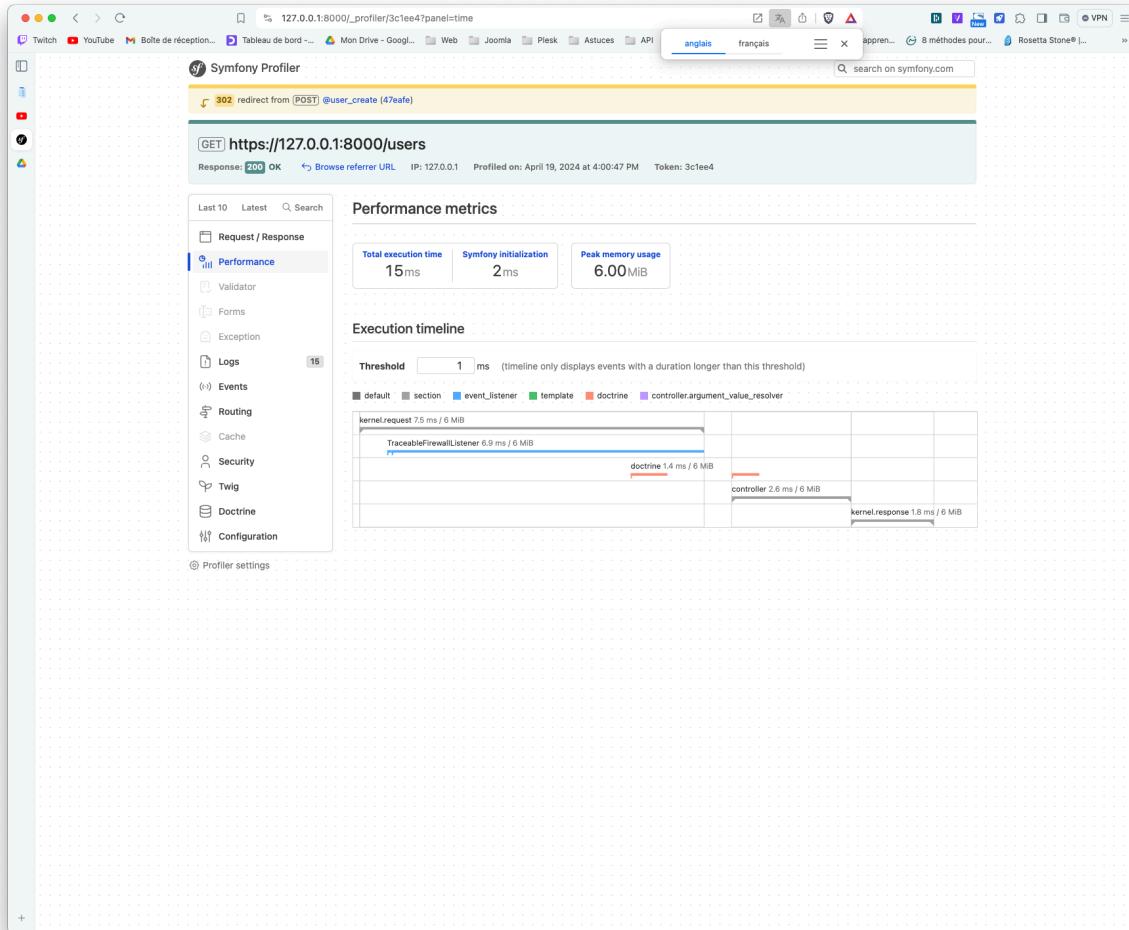
User create



User edit



Users after create



Users after edit

