Audit de la dette technique et de la performance

Florian Jourde

Mars 2023

Table des matières

1. Introduction

Mars 2023

Présentation du projet

Le client **ToDo & Co** souhaite améliorer son application de prise de notes, de manière à pérenniser son système sur le long terme, optimiser la performance et ainsi pouvoir se développer.

Une première version de ce site web ayant déjà été développé en **2016**, la solution technique s'en retrouve aujourd'hui daté. Nous devons donc identifier les points faibles de l'application afin d'assurer une stabilité de l'application sur le **long terme**.

À travers cet audit, nous allons cibler ensemble les axes d'améliorations pertinents, et ceux dont nous pouvons faire abstraction.

Outils utilisés

Pour réaliser cet audit, plusieurs outils à la disposition du développeur vont être nécessaire, afin d'évaluer l'efficacité de l'application.

Parmi ces outils nous retrouvons :

- **PhpStorm**, dans sa version **2021.1.4**, environnement de développement professionnel ;
- **WampServer**, dans sa version **1.2.6**, serveur local permettant de basculer entre les différentes version de PHP ;
- **PHP 5.6** & **PHP 8.1**, versions du langage portant le même nom, majoritairement utilisés sur le web ;
- **Symfony 3.1**, **Symfony 5.4**, framework PHP majoritairement répandu, permettant la simplification de la prise en main du code de l'application, nativement optimisé pour la sécurité ;
- **Symfony Profiler**, déboguer embarqué par Symfony, qui nous permettra d'évaluer la performance et la mémoire globale utilisé par l'application ;
- **PHPUnit 9.5**, bibliothèque PHP très répandue, permettant, entre-autres, de réaliser des tests unitaires et fonctionnels ;

- Le navigateur **Google Chrome**, qui nous permettra de tester l'application ;
- L'outil **DevTools** de ce même navigateur, dit « console », qui nous servira à détecter les anomalies liées à plusieurs aspect du web : performance, affichage, fonctionnalités...

Méthodologie adoptée

La méthodologie de développement adoptée pour l'évolution de cette application sera de type **Domain Driven Design**. De cette manière, nous nous assurerons que toutes les améliorations apportées soient en adéquations avec le résultat attendu par le client.

Aussi, pour prendre en main l'application, apprivoiser ses subtilités, et donc pouvoir proposer au client **ToDo & Co** l'amélioration la plus pertinente possible, le courant audit a été réalisé en **fin de phase de développement** du projet.

2. Audit de la dette technique

Mars 2023

Solution technique initiale

La première étape à réaliser pour débuter l'évolution de l'application de prise de notes **ToDo & Co** va être de prendre connaissance du travail effectué en amont sur l'application.

L'application initiale a été prise en charge par **saro0h**. Vous pourrez trouver le code de la précédente version de cette application ici :

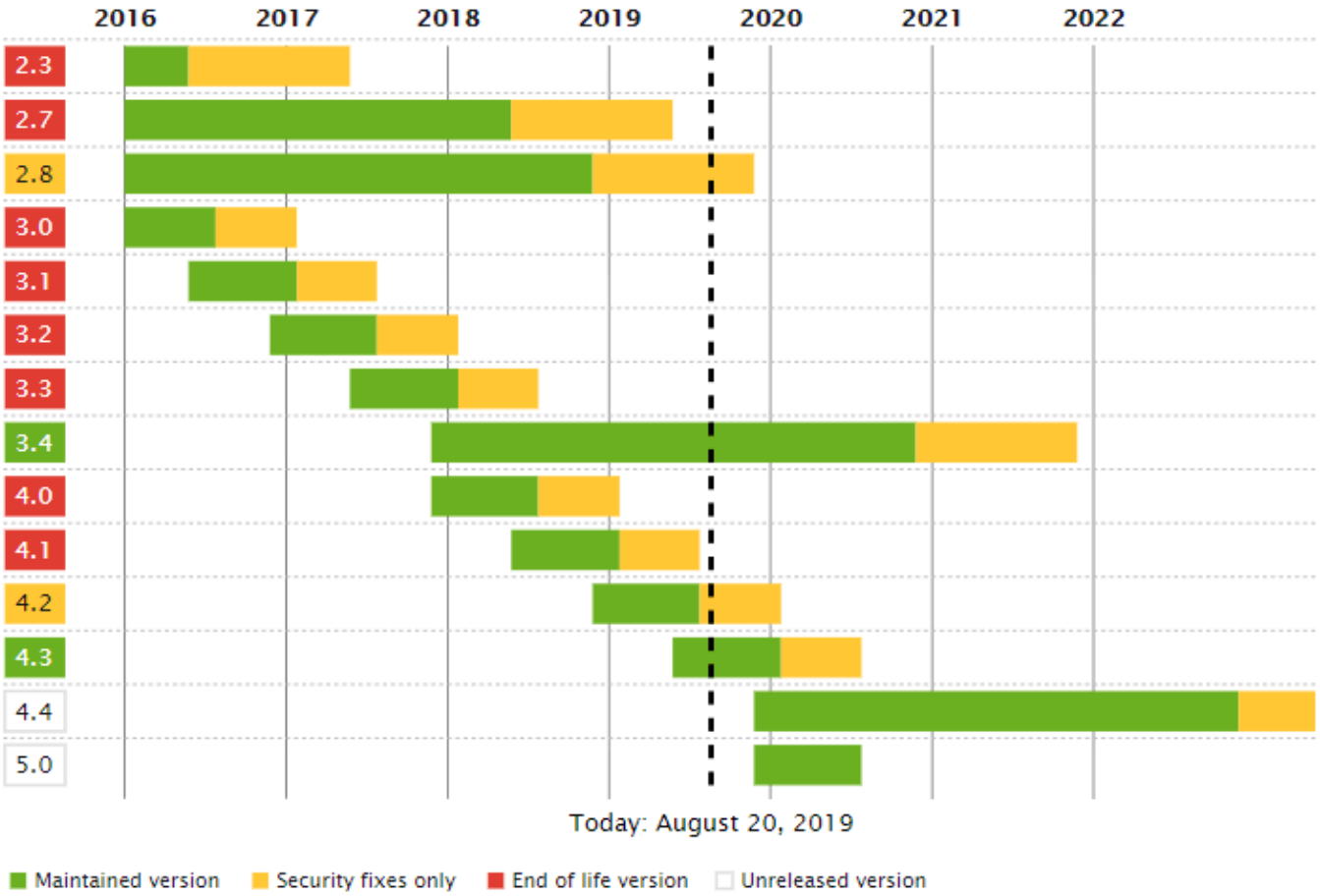
https://github.com/saro0h/projet8-TodoList

Après une première prise de connaissance du projet, nous nous apercevons que l'application a été développé sur **Symfony 3.1**.

```
19      "require": {
20          "php": "⩾5.5.9",
21          "symfony/symfony": "3.1.*", v3.1.6
```

Un rapide test nous permet de nous apercevoir que l'application fonctionne correctement sous **PHP 5.6**. Les entités ayant correctement été conçues, la base de données est fonctionnelle et les informations transitent bien.

Cependant, le principal problème avec cette solution technique est sa durée de vie. La version de **Symfony 3.1** ayant été maintenue de **2016 à 2017,** comme l'indique le tableau ci-après, elle peut aujourd'hui présenter des **failles de sécurité** majeures.



À titre d'exemple, nous pouvons constater que la méthodologie d'accès à certaines dépendances de Symfony a varié, au fil du temps.

C'est notamment le cas de cette dépendance, nommée « **getDoctrine()** », qui s'est vue attribuer un nouveau fonctionnement.

```
17     public function listAction()
18     {
19         return $this->render( view: 'task/list.html.twig', ['tasks' =>
       $this->getDoctrine()->getRepository('AppBundle:Task')->findAll()]);
20     }
```

Laisser une dépréciation de ce type dans le code n'est pas recommandé par les créateurs du framework.

Toujours à propos de ces failles de sécurité, voici une nouvelle dépréciation qu'il n'est pas convenable de laisser telle quelle.

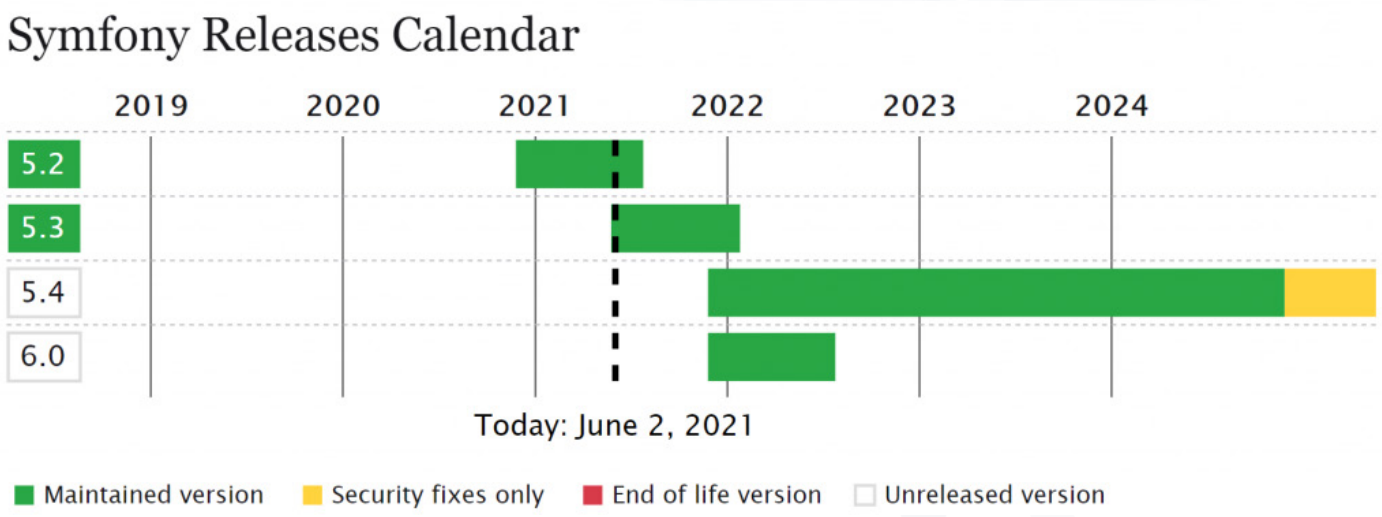
```
17        $authenticationUtils = $this->get('security.authentication_utils');
```

Solution technique proposée

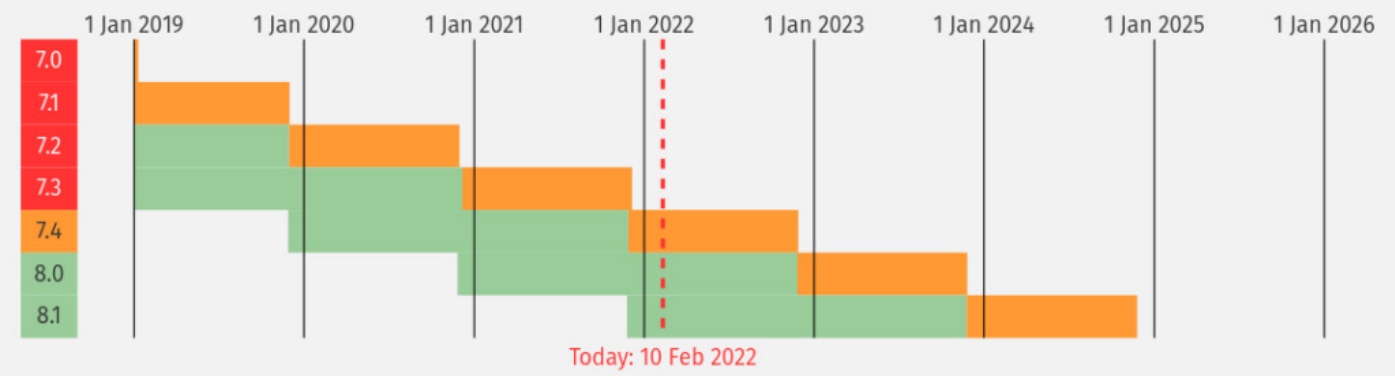
Après avoir tenté, sans succès, de mettre à jour la version de Symfony, ainsi que celle de PHP, la meilleure option à envisager semble être celle d'une **migration** de l'application, plutôt qu'une mise à jour.

En effet, les **cœurs** sur lesquels se basent le langage PHP et le framework Symfony ont tellement évolué, qu'il serait **contre-productif** de tenter d'effectuer les mises à jour une à une.

Étant donné l'**envergure modérée** de l'application, il sera plus pertinent de réaliser une refonte complète de **ToDo & Co**, en se basant sur les versions **Long-Term Support** (LTS) de PHP & Symfony, dont voici les durée de vie :



Et voici le cycle de vie de PHP :



Pour cette raison, la solution technique à privilégier pour **ToDo & Co** sera **Symfony LTS 5.4** & **PHP 8.1**.

Qualité de code

Le framework Symfony nous permet de récupérer le code développé par **saro0h** (public, src, templates...), ce qui est un gain de temps important. Grâce au système d'**entités**, nous n'avons pas non plus à nous soucier de la **migration** de la base de données.

Ensuite, la version la plus récente de PHP étant fonctionnelle avec le projet, il sera aisé pour les développeurs en charge du développement futur de l'application, d'adapter le code aux nouvelles normes préconisées par PHP ou Symfony, comme par exemple, l'évolution vers le nouveau système d'**attributs** de **PHP 8.**

Sécurité

L'évolution du **système d'authentification** ayant été réalisé (cf. PDF dédié) et la migration vers **Symfony 5.4** effectuée, nous n'avons désormais plus à nous inquiéter des failles majeures présentes sur l'application initiale.

3. Audit de la performance

Mars 2023

Performances initiales

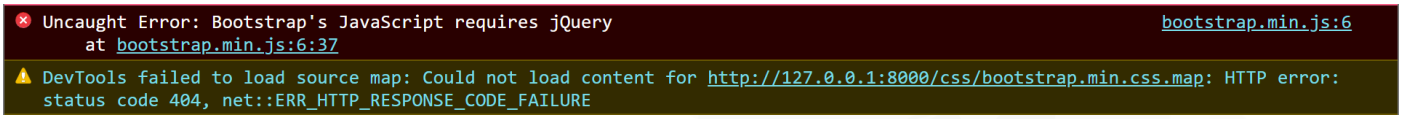
La question de la performance sur le projet est moins prépondérante que celle de la qualité de code.

En effet, le projet initial étant relativement léger, aucun problème de performance majeur n'a été identifié, seulement des axes d'améliorations.

Parmi ces axes d'améliorations, l'un des points ayant été détectés concerne la présence de **Boostrap** & **jQuery**. Ces bibliothèques peuvent grandement faciliter le dynamisme et la rapidité d'intégration lors de développement conséquents, leur utilisation s'avère un peu exagérée, pour un projet de ce type.

```
94          <script src="{{ asset('js/jquery.js') }}"></script>
95          <script src="{{ asset('js/bootstrap.min.js') }}"></script>
```

De plus, des erreurs s'affichent en console, indiquant que **jQuery** ne peut être chargé. Une sollicitation de mémoire vive dont on peut aisément se passer a donc lieu.

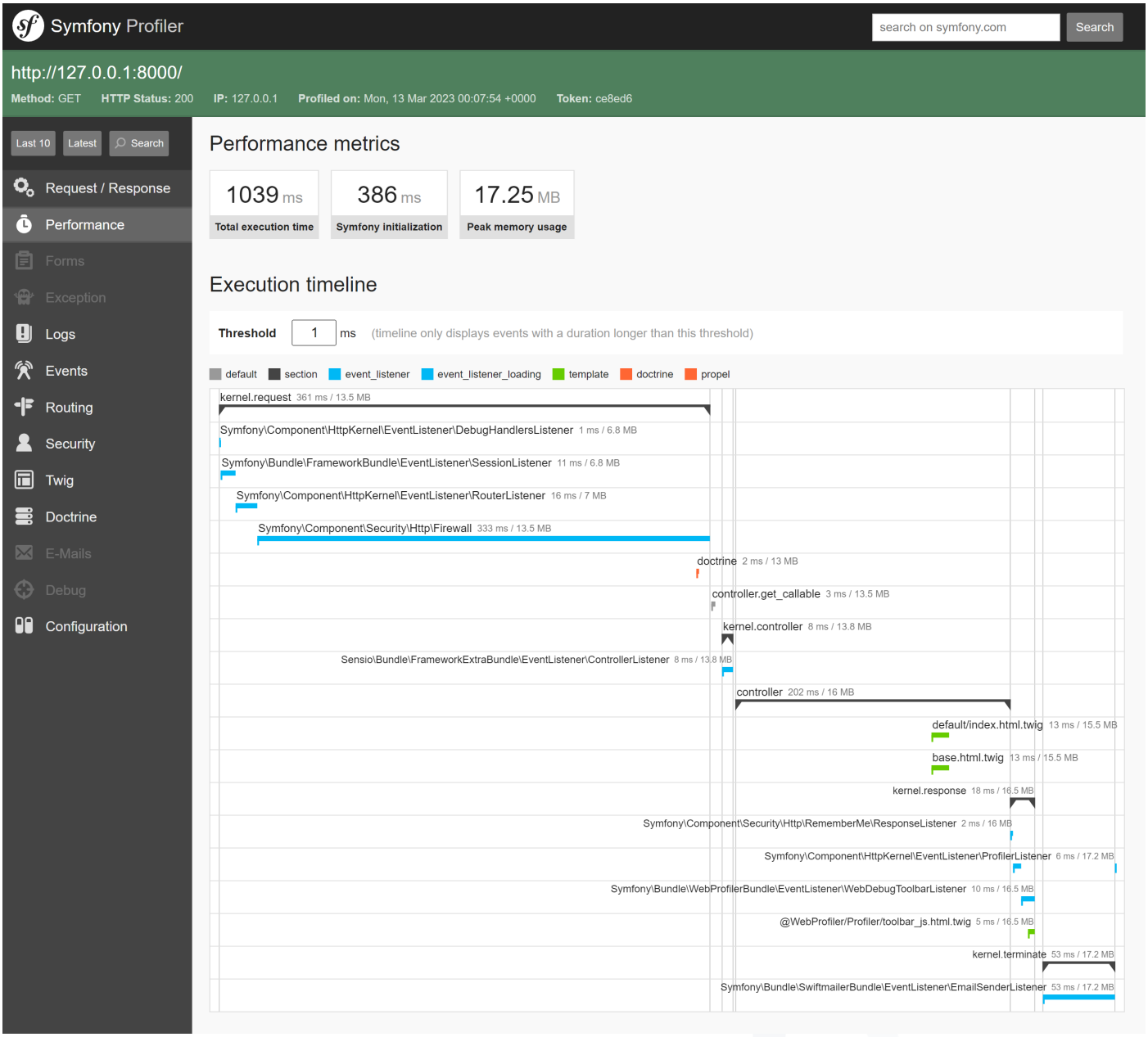


Une fois les fichiers liés à ces deux bibliothèques et leurs appel dans le code supprimés, nous nous retrouvons avec une application plus performante.

Cette action n'est pas indispensable, mais vivement recommandée dans le cas présent, étant donnée que c'est la **performance** qui est recherchée pour l'application **ToDo & Co**.

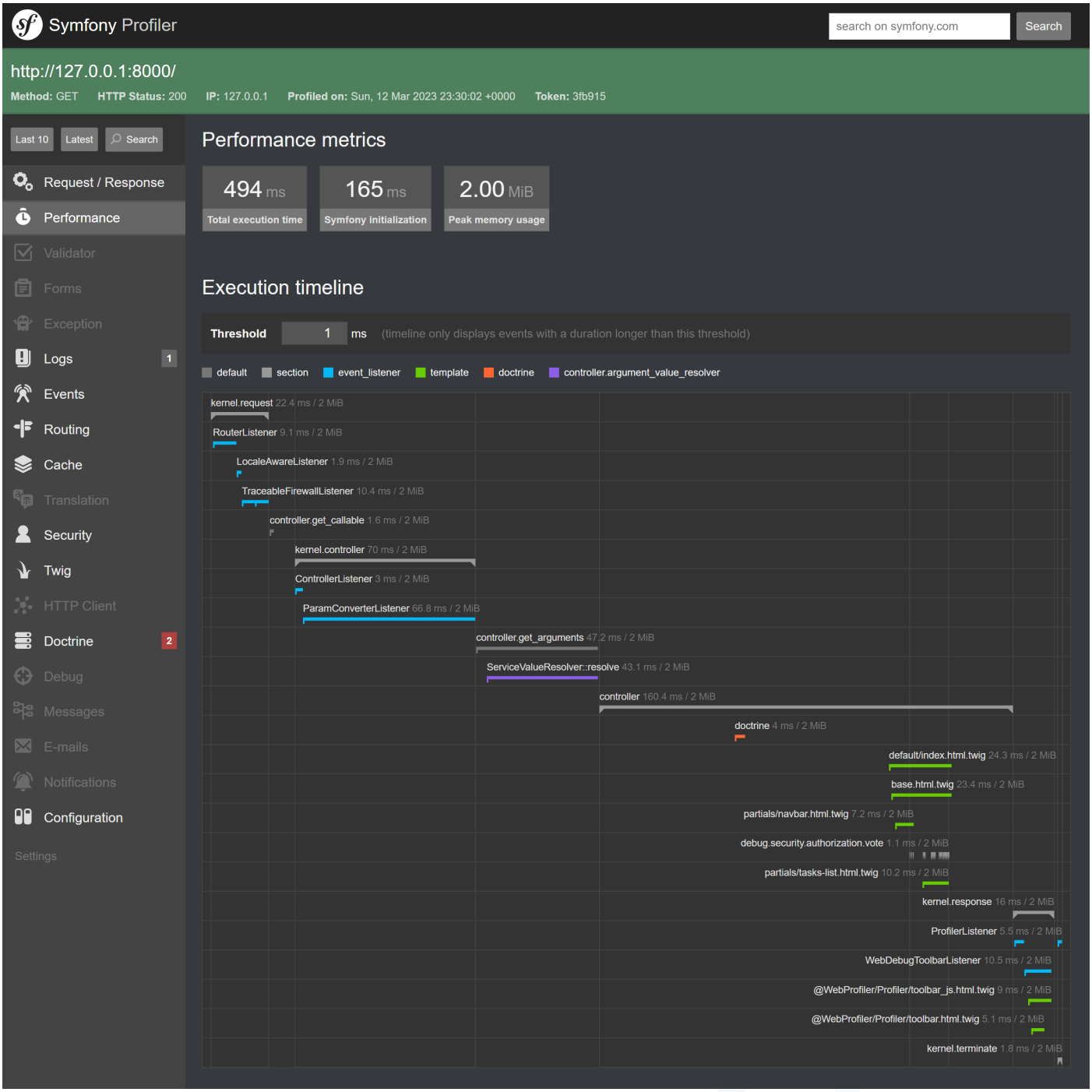
Un temps devra cependant être alloué à la réalisation de **feuilles de styles**, afin de garder un affichage et une compréhension de l'interface utilisateur convenable.

La capture d'écran suivante présente les performances de la page d'accueil de Symfony. Ces données sont délicates à interpréter sans **outils de diagnostics** payants (cf. **Blackfire**), mais on peut tout de même constater que les performances sont anormalement élevées pour une simple page, ne faisant appel à aucune entité en base de données : le jeu de données n'était pas initialisé lors de ce test.



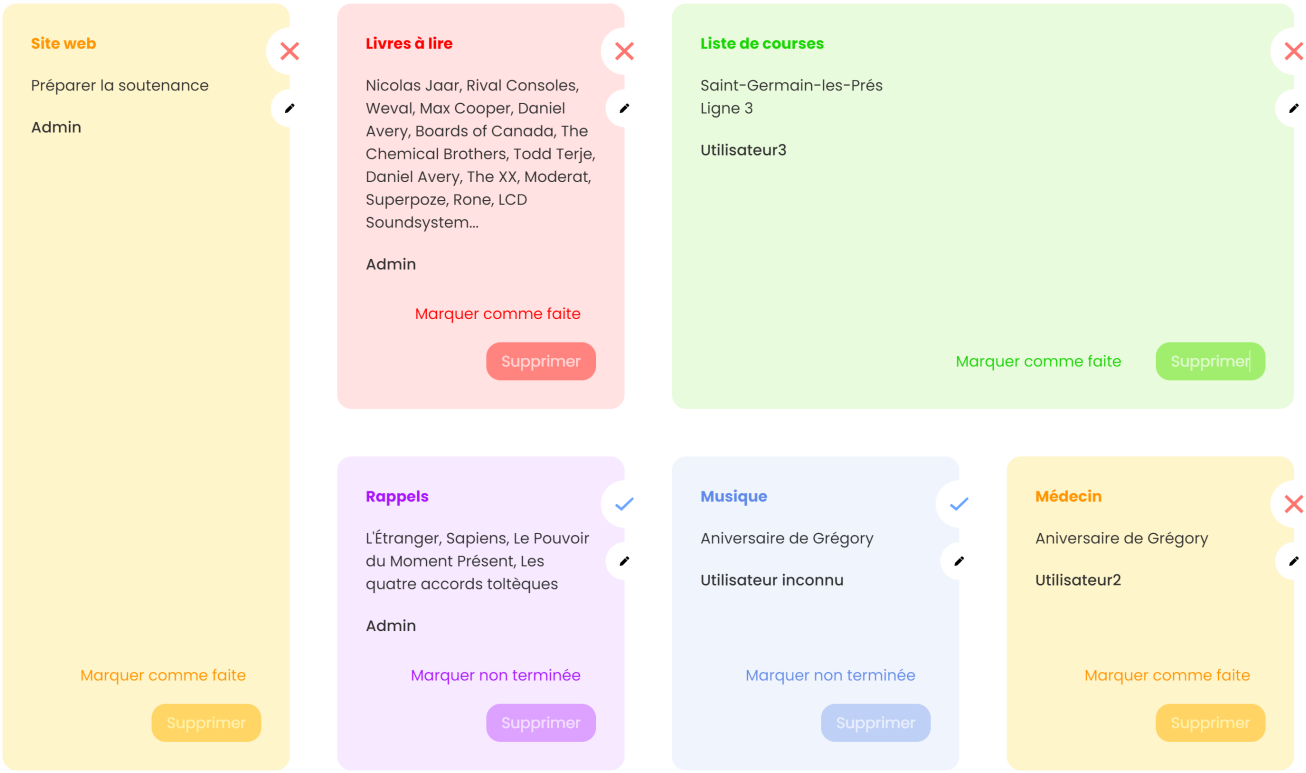
Performances finales

Nous pouvons constater que le simple fait de basculer sur la version **Symfony 5.4** et **PHP 8.1** a amélioré les performances.



Ici, le jeu de données avait pourtant bel et bien été inclus !

Le chargement de la page a été diminué par trois, le pic de mémoire vive a diminué de près de 10 fois ! Le gain de performance obtenu n'est donc pas négligeable.



Au delà du gain de performance obtenu par la suppression de **Bootstrap** & **jQuery**, la souplesse d'un **Design System** conçu en **CSS** natif augmente considérable l'expérience utilisateur.