

To Do List App

Rapport d'audit



Table des matières

Contexte.....	2
Outils d'analyse utilisés	2
Analyse de l'existant	3
1. Framework Symfony	3
2. PHP	3
3. Les anomalies	3
a. Les anomalies rapportées	3
b. Les anomalies identifiées	4
4. Analyse du code.....	4
b. PHP Stan	5
5. Analyse de performance	7
Solutions apportées	8
1. Mise à jour de Symfony, PHP et Bootstrap.....	8
2. Correction des anomalies	8
a. Correction des anomalies rapportées	8
b. Correction des anomalies identifiées	8
3. Ajout de nouvelles fonctionnalités.....	9
a. Autorisation	9
b. Implémentation des tests automatisés	10
c. Intégration continue	11
4. Analyse de performance	12

Contexte

ToDo & Co est une startup dont le cœur de métier est l'application **ToDoList** permettant de gérer ses tâches quotidiennes. Afin de montrer à de potentiels investisseurs que le concept est viable, l'application a été développée sous forme MVP (Minimum Viable Product en anglais ou Produit minimum Viable en français).

La startup a réussi à lever des fonds pour permettre le développement de l'entreprise et de l'application. L'amélioration de la qualité de l'application est alors nécessaire de par l'implémentation de nouvelles fonctionnalités et de tests automatisés ainsi que la correction d'anomalies.

Pour cela, une analyse du projet initial, sur sa qualité et ses performances, est donc nécessaire pour définir les besoins éventuels de mise à jour et d'amélioration.

Outils d'analyse utilisés

	Outils	Description
Qualité	PHPStan	Outil d'analyse statique PHP pour détecter les erreurs
	PHP CS Fixer	Outil de correction de la syntaxe PHP
	Symfony Insight	Outil d'analyse en ligne du code PHP spécialisé Symfony
	Lignes de commandes incluses avec Symfony et composer	Analyse des fichiers Twig, des fichiers de configuration, ...
Performance	Profiler de Symfony	Collecte des informations sur l'état de l'application à l'exécution d'une requête

Analyse de l'existant

1. Framework Symfony

L'application a été développée avec la version 3.1 du framework Symfony. Cette version n'est plus maintenue depuis janvier 2017 pour les bugs et depuis juillet 2017 pour la sécurité. Les composants Symfony et les bibliothèques tierces ne sont également plus à jour voir sont dépréciés.

Il est donc nécessaire d'effectuer la mise à jour du framework vers la version actuelle qui est la 6.2. De par la suite, l'application pourra évoluer vers la version 6.4 du framework, prévu en novembre 2023, qui sera une version LTS (Long-Term Support) supportée trois ans et quatre ans pour la sécurité.

2. PHP

Concernant la version de PHP, le projet utilise la version 5.6.4 qui est actuellement obsolète, l'application s'expose donc à des risques de sécurité. Le projet doit passer à la dernière version de PHP qui est la 8.2 sortie en décembre 2022.

3. Les anomalies

a. Les anomalies rapportées

Ci-dessous la liste des anomalies rapportées par la startup :

- Une tâche doit être rattachée à l'utilisateur authentifié lors de sa création.
- Lors de la modification d'une tâche, l'auteur n'est pas modifiable.
- Les tâches déjà créées doivent être rattachées à un utilisateur « anonyme ».
- Choisir un rôle pour un utilisateur (administrateur ou utilisateur) lors de la création et de la modification d'un compte.

b. Les anomalies identifiées

Après examen du projet, d'autres anomalies ont pu être identifiées :

- Lien vers la page d'accueil inexistant
- Lien vers la page liste des utilisateurs inexistant
- Page liste des tâches terminées inexistante
- Page liste des tâches à faire comprend toutes les tâches
- Message flash indiquant uniquement tâche marquée terminée
- Manque la gestion du pseudo unique
- Absence validation mot de passe
- Formulaire et message d'erreur de création et édition de tâche en anglais
- Absence des pages erreur (404 et 403) en production
- Absence de token dans le formulaire de connexion (faille CSRF)
- Absence de JQuery pour l'utilisation de Bootstrap
- Absence des méthodes HTTP pour les différentes routes

b. PHP Stan

PHP Stan qui est un analyseur statique pour PHP, permet de capturer les erreurs du niveau le plus souple au plus strict. Pour ce projet, le niveau 9 qui est le plus haut a été utilisé pour avoir une analyse la plus strict.

```

PS C:\laragon\www\P8_ToDoList> vendor/bin/phpstan analyse -l 9 src tests
10/10 [=====] 100%

-----
Line    src\AppBundle\Controller\DefaultController.php
-----
:13     Method AppBundle\Controller\DefaultController::indexAction() has no return type specified.

-----
Line    src\AppBundle\Controller\SecurityController.php
-----
:14     Method AppBundle\Controller\SecurityController::loginAction() has no return type specified.
:18     Call to an undefined method object::getLastAuthenticationError().
:19     Call to an undefined method object::getLastUsername().
:30     Method AppBundle\Controller\SecurityController::loginCheck() has no return type specified.
:38     Method AppBundle\Controller\SecurityController::logoutCheck() has no return type specified.

-----
Line    src\AppBundle\Controller\TaskController.php
-----
:16     Method AppBundle\Controller\TaskController::listAction() has no return type specified.
:18     Call to method getRepository() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:24     Method AppBundle\Controller\TaskController::createAction() has no return type specified.
:32     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:48     Method AppBundle\Controller\TaskController::editAction() has no return type specified.
:55     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:71     Method AppBundle\Controller\TaskController::toggleTaskAction() has no return type specified.
:74     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:84     Method AppBundle\Controller\TaskController::deleteTaskAction() has no return type specified.
:86     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols

-----
Line    src\AppBundle\Controller\UserController.php
-----
:16     Method AppBundle\Controller\UserController::listAction() has no return type specified.
:18     Call to method getRepository() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:24     Method AppBundle\Controller\UserController::createAction() has no return type specified.
:32     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols
:33     Call to an undefined method object::encodePassword().
:50     Method AppBundle\Controller\UserController::editAction() has no return type specified.
:57     Call to an undefined method object::encodePassword().
:60     Call to method getManager() on an unknown class Doctrine\Persistence\ManagerRegistry.
       📖 Learn more at https://phpstan.org/user-guide/discovering-symbols

-----
Line    src\AppBundle\Entity\Task.php
-----
:19     Property AppBundle\Entity\Task::$id has no type specified.
:19     Property AppBundle\Entity\Task::$id is never written, only read.
       📖 See: https://phpstan.org/developing-extensions/always-read-written-properties
:24     Property AppBundle\Entity\Task::$createdAt has no type specified.
:30     Property AppBundle\Entity\Task::$title has no type specified.
:36     Property AppBundle\Entity\Task::$content has no type specified.
:41     Property AppBundle\Entity\Task::$isDone has no type specified.
:49     Method AppBundle\Entity\Task::getId() has no return type specified.
:54     Method AppBundle\Entity\Task::getCreatedAt() has no return type specified.
:59     Method AppBundle\Entity\Task::setCreatedAt() has no return type specified.
:59     Method AppBundle\Entity\Task::setCreatedAt() has parameter $createdAt with no type specified.
:64     Method AppBundle\Entity\Task::getTitle() has no return type specified.
:69     Method AppBundle\Entity\Task::setTitle() has no return type specified.
:69     Method AppBundle\Entity\Task::setTitle() has parameter $title with no type specified.
:74     Method AppBundle\Entity\Task::getContent() has no return type specified.
:79     Method AppBundle\Entity\Task::setContent() has no return type specified.
:79     Method AppBundle\Entity\Task::setContent() has parameter $content with no type specified.
:84     Method AppBundle\Entity\Task::isDone() has no return type specified.
       📖 See: https://phpstan.org/blog/solving-phpstan-no-value-type-specified-in-iterable-type

-----
Line    tests\AppBundle\Controller\DefaultControllerTest.php
-----
:19     Method Tests\AppBundle\Controller\DefaultControllerTest::testIndex() has no return type specified.
:15     Call to an undefined method Tests\AppBundle\Controller\DefaultControllerTest::assertEquals().
:15     Cannot call method getStatusCode() on Symfony\Component\HttpFoundation\Response|null.
:16     Call to an undefined method Tests\AppBundle\Controller\DefaultControllerTest::assertContains().

[ERROR] Found 67 errors

```

PHP Stan indique que le projet présente 67 erreurs de code PHP qui concernent essentiellement des erreurs de typage pour les retours de méthodes, les paramètres dans les fonctions et les attributs des entités.

5. Analyse de performance

L'ensemble des données d'analyse ont été regroupées dans un fichier Excel accessible depuis le dossier « documents » à la racine du projet.

Le profiler de Symfony a été utilisé pour l'analyse de performance de l'application. Pour mesurer la performance, 3 catégories de données ont été collectées :

- **Performance metrics** : temps total d'exécution, temps d'initialisation de Symfony, quantité de mémoire allouée
- **Twig metrics** : temps pour l'affichage de la vue, nombre de bloc Twig
- **Doctrine metrics** : nombre de requête SQL exécuté, temps d'exécution

Ci-dessous un aperçu du fichier Excel regroupant les données des différentes routes (GET et POST) :

URI	Nom route	Méthode HTTP	Performance Metrics			Twig Metrics			Query Metrics	
			Total execution time (ms)	Symfony initialization (ms)	Peak memory usage (MiB)	Render time (ms)	Template calls	Block calls	Database Queries	Query time (ms)
/	homepage	GET	225	34	2	7	5	3	1	0,67
/login	login	GET	242	67	10	13	4	3	0	0
POST - /login_check	login_check	POST	530	34	2	0	0	0	1	1,11
/tasks	task_list	GET	195	33	2	5	5	3	2	1,4
/tasks/create	task_create	GET	351	32	6	20	5	39	1	1,05
POST - /tasks/create	task_create	POST	304	46	2	0	0	0	4	3,19
/tasks/{id}/edit	task_edit	GET	316	45	6	17	5	39	2	1,25
POST - /tasks/{id}/edit	task_edit	POST	267	34	4	0	0	0	5	3,85
/users	user_list	GET	257	41	2	8	5	3	1	0,4
/users/create	user_create	GET	463	64	8	227	4	62	0	0
POST - /users/create	user_create	POST	1414	44	2	0	0	0	4	22
/users/{id}/edit	user_edit	GET	667	32	10	413	5	62	1	0,74
POST - /users/{id}/edit	user_edit	POST	945	32	4	0	0	0	5	7,66

On peut remarquer un traitement long des formulaires, notamment dû au système de hachage de mot de passe et à la validation des différentes données soumises. Il faut noter également que le temps d'exécution des pages liste des tâches et utilisateurs augmentera quand le nombre de données à afficher sera plus important. Un système de pagination peut être mis en place pour éviter cela.

Solutions apportées

1. Mise à jour de Symfony, PHP et Bootstrap

Comme indiqué précédemment, l'application a évolué vers la dernière version 6.2 de Symfony qui pourra par la suite évoluer plus facilement vers la version 6.4 LTS prévu en novembre 2023. Les dépendances sont alors également mises à jour. La version de PHP a également été mise à jour vers la version 8.2.

Concernant la partie Front-End, la version de Bootstrap a été mise à jour (3.3.7 vers 5.0.1). Cette version de Bootstrap n'utilise plus JQuery pour son fonctionnement ce qui entraîne donc un gain de temps pour l'affichage des pages.

2. Correction des anomalies

a. Correction des anomalies rapportées

La liste des anomalies rapportées a été corrigée. A savoir :

- Une tâche est rattachée à l'utilisateur authentifié lors de sa création.
- Lors de la modification d'une tâche, l'auteur n'est pas modifiable.
- Les tâches déjà créées sont rattachées à un utilisateur « anonyme ».
- Le choix d'un rôle pour un utilisateur (administrateur ou utilisateur) lors de la création et de la modification d'un compte est possible.

b. Correction des anomalies identifiées

Les différentes anomalies identifiées ont été corrigées :

- Ajout du lien vers la page d'accueil
- Ajout d'un lien vers la page liste des utilisateurs (uniquement administrateur)
- Création de la page liste des tâches terminées
- Modification de la page liste des tâches à faire

- Ajout d'un message flash pour le marquage d'une tâche non terminée
- Gestion du pseudo unique lors de la création d'un utilisateur
- Formulaire et message d'erreur de création et édition de tâche traduit en français
- Création des pages erreur (404 et 403)
- Ajout d'un token dans le formulaire de connexion
- Ajout des méthodes HTTP pour les différentes routes

Un refactoring a été également effectué en mettant en place un service de gestion des formulaires lors de la soumission. Ce qui permet de décharger les Controller, d'avoir une meilleure compréhension du code et surtout d'avoir un code réutilisable.

3. Ajout de nouvelles fonctionnalités

a. Autorisation

Les fonctionnalités souhaitées suivantes ont été mises en place :

- Seuls les utilisateurs ayant le rôle administrateur doivent accéder aux pages de gestions 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

La gestion d'accès aux différentes pages de l'application, a été réalisée dans le fichier de configuration **security.yaml**.

Un système de Voter a été mis en place pour la suppression des tâches uniquement par l'auteur et la suppression des tâches anonyme par l'administrateur.















b. Implémentation des tests automatisés

Des tests unitaires et fonctionnels pour assurer le fonctionnement de l'application ont été implémentés avec PHPUnit.

Tests unitaires

Des tests unitaires ont été réalisés sur les entités. Il s'agit principalement des tests sur les contraintes de validation des données.














Couverture de code de l'entité Task :

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	17 / 17		100.00%	12 / 12			100.00% 1 / 1
Task		100.00%	17 / 17		100.00%	12 / 12	12		100.00% 1 / 1
__construct		100.00%	2 / 2		100.00%	1 / 1	1		
getId		100.00%	1 / 1		100.00%	1 / 1	1		
getCreatedAt		100.00%	1 / 1		100.00%	1 / 1	1		
setCreatedAt		100.00%	2 / 2		100.00%	1 / 1	1		
getTitle		100.00%	1 / 1		100.00%	1 / 1	1		
setTitle		100.00%	2 / 2		100.00%	1 / 1	1		
getContent		100.00%	1 / 1		100.00%	1 / 1	1		
setContent		100.00%	2 / 2		100.00%	1 / 1	1		
isDone		100.00%	1 / 1		100.00%	1 / 1	1		
toggle		100.00%	1 / 1		100.00%	1 / 1	1		
getAuthor		100.00%	1 / 1		100.00%	1 / 1	1		
setAuthor		100.00%	2 / 2		100.00%	1 / 1	1		

Tests Fonctionnels

Les tests fonctionnels ont pour but de garantir que le comportement fonctionnel obtenu est bien conforme avec celui attendu.

Couverture de code du Controller UserController :

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	12 / 12		100.00%	3 / 3			100.00% 1 / 1
UserController		100.00%	12 / 12		100.00%	3 / 3	5		100.00% 1 / 1
listAction		100.00%	1 / 1		100.00%	1 / 1	1		
register		100.00%	7 / 7		100.00%	1 / 1	2		
editAction		100.00%	4 / 4		100.00%	1 / 1	2		

Résultat du rapport de couverture de code

Au total, 56 tests implémentés avec 111 assertions

```
PS C:\laragon\www\P8_ToDoList> vendor/bin/phpunit
PHPUnit 10.1.2 by Sebastian Bergmann and contributors.

























Runtime:       PHP 8.2.4
Configuration: C:\laragon\www\P8_ToDoList\phpunit.xml.dist

.....                                     56 / 56 (100%)

Time: 00:09.477, Memory: 56.00 MB

OK (56 tests, 111 assertions)
```

Le taux de couverture est supérieur à 80% (minimum 70% souhaité)

	Code Coverage							
	Lines		Functions and Methods			Classes and Traits		
Total		80.25% 191 / 238		84.06% 58 / 69		68.42% 13 / 19		
■ Controller		94.44% 51 / 54		76.92% 10 / 13		50.00% 2 / 4		
■ DataFixtures		0.00% 0 / 26		0.00% 0 / 2		0.00% 0 / 1		
■ Entity		100.00% 34 / 34		100.00% 23 / 23		100.00% 2 / 2		
■ Form		100.00% 35 / 35		100.00% 2 / 2		100.00% 2 / 2		
■ Handlers		100.00% 58 / 58		100.00% 18 / 18		100.00% 7 / 7		
■ Repository		11.11% 2 / 18		28.57% 2 / 7		0.00% 0 / 2		
■ Security		84.62% 11 / 13		75.00% 3 / 4		0.00% 0 / 1		

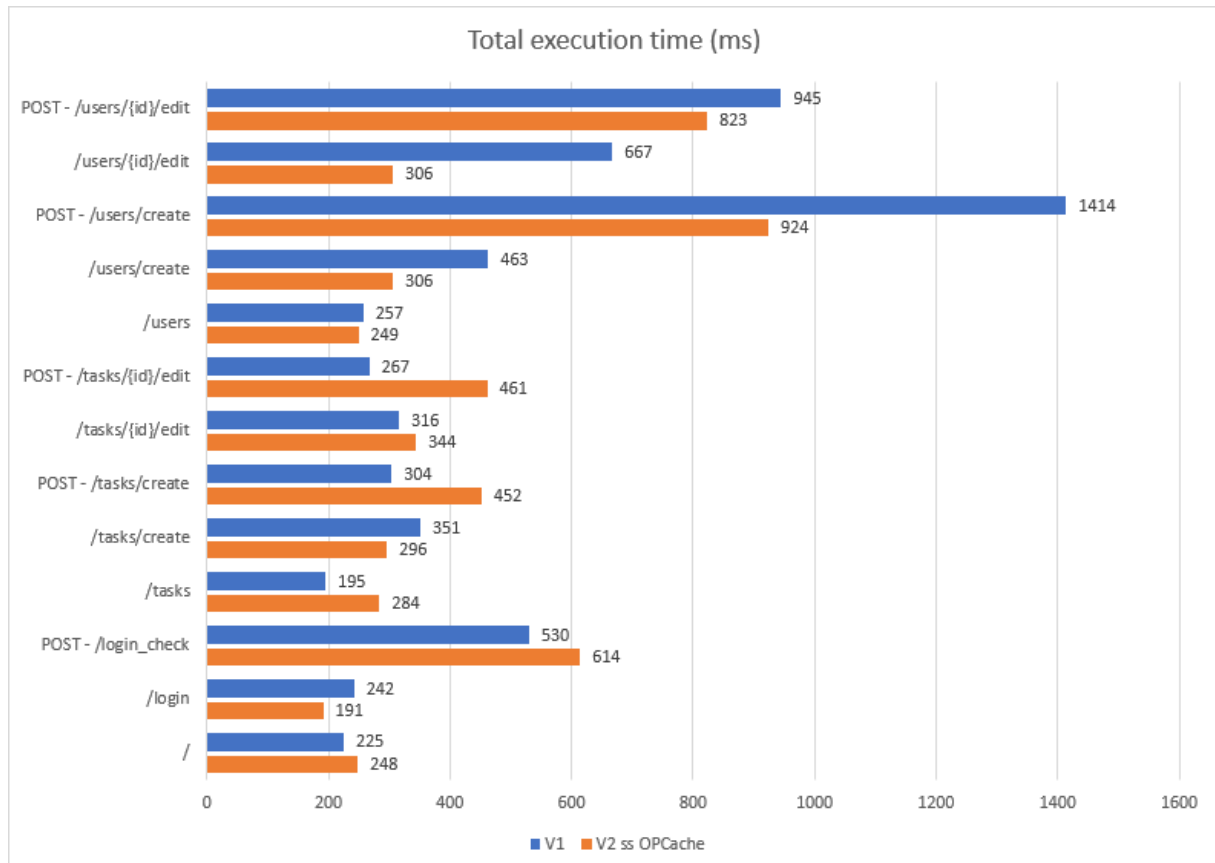
Le rapport de couverture est accessible depuis le dossier **public/test-coverage** au format HTML.

c. Intégration continue

Un outil d'intégration continue a été mis en place sur le dépôt GitHub grâce à Github Workflow. L'outil permet de lancer l'application et l'ensemble des tests unitaires et fonctionnels ainsi qu'une analyse PHPStan pour s'assurer qu'à chaque PUSH sur le dépôt GitHub l'application est fonctionnelle. Ce qui permet de valider un changement et donc d'avoir une version exécutable en production.

4. Analyse de performance

Ci-dessous un comparatif du temps total d'exécution pour chaque route entre la version initiale et la version final du projet :

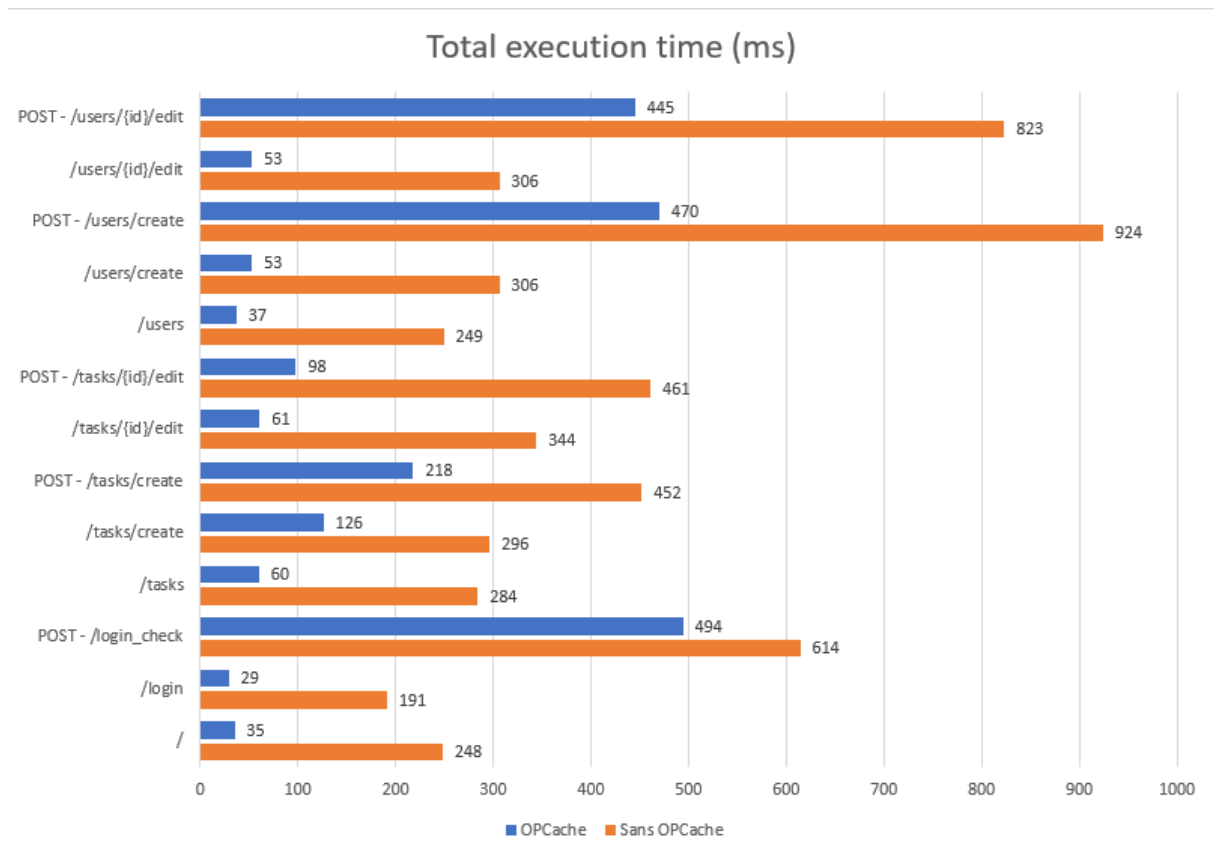


On peut remarquer globalement que le temps d'exécution est réduit pour les différentes page de l'application.

Néanmoins, il est encore possible de réduire ce temps en activant l'extension PHP OPCache sur le serveur de l'application.

OPCache améliore les performances de PHP. Il n'est donc plus nécessaire à PHP de charger et d'analyser les scripts à chaque requête.

Voici un comparatif du total temps d'exécution de la version final de l'application avec et sans OPCache :



On remarque que le total temps d'exécution est réduit jusqu'à 7 fois. L'application est alors plus rapide.