Rapport d’audit de qualité de code et de performance

Réalisation : Jean-Eudes Nouaille-Degorce

Symfony
5.2

PLAN

Améliorations majeures

Analyses de la qualité des codes

Rapports de performances

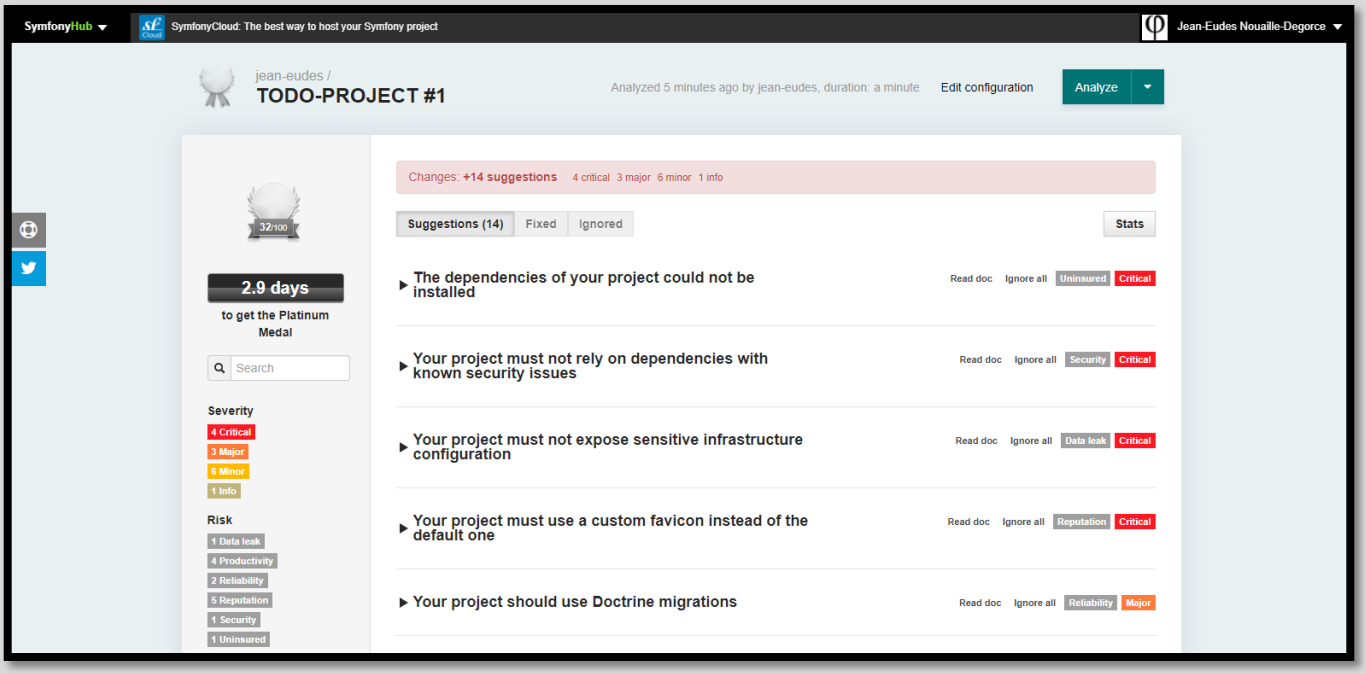
1

Améliorations majeures

Modification de la version de Symfony & mise à jour des dépendances

- Au départ, l'application utilisait la version 3.1 de Symfony qui n'est plus supportée. Une migration a été effectuée vers la version 5.2 afin de résoudre plusieurs erreurs critiques.

- Une première analyse Symfony Insight du projet initial a en effet permis de détecter des dépréciations majeures, notamment l'impossibilité d'installer certaines dépendances dont les versions ne sont plus à jour. Les tentatives d'update via Composer du projet se sont soldées par des messages d'erreurs : en raison d'incompatibilités entre les différentes versions des dépendances.



Améliorations des templates

- Mise à jour du formulaire de connexion avec une case à cocher pour autoriser les cookies.
- Ajouts des titres des pages qui manquaient.
- Mise en place d'un affichage alternatif entre la liste de toutes les tâches et la liste des tâches terminées (cf. fonction « listDone » ajoutée au fichier TaskController).
- Ajout de boutons de suppressions dans le dashboard utilisateurs (accessible uniquement aux administrateurs).
- Filtrage de l'affichage des boutons de suppressions sur la liste des tâches en fonction des rôles accordés aux utilisateurs.
- Ajouts de boutons de pagination.
- Ajouts d'icônes « Glyhicons » pour améliorer la lisibilité des boutons.
- Ajouts d'un menu modal Bootstrap réservé aux administrateurs pour améliorer la navigation sur le site.
- Révisions de l'appel des dépendances CSS et JS avec Webpack Encore.
- Responsive design mis en place sur le dashboard utilisateurs et sur la page des tâches.
- Retouches CSS multiples : centrages de contenus, colorations, ombres légères, tailles des textes…

Améliorations des controllers

- Ajouts de constructeurs sur les classes « UserController » et « TaskController ».
- Révisions des appels des fichiers « /Repository ».
- Ajouts de la pagination via un service « PagingHandler ».
- Filtrages de la liste des tâches par dates de créations ou dates de mises à jour.
- Ajouts de messages Flash pour notifier les tâches marquées terminées ou en cours.
- Envois de messages d'erreurs lors de la gestion des tâches ou lors de la gestion des utilisateurs (erreurs générées par les fichiers « UserVoter » et « TaskVoter »).

- Ajouts des paramètres de retour et des méthodes utilisées pour chaque fonction des controllers (GET, POST, DELETE) :

```
homepage        GET         ANY     ANY     /
login           GET|POST    ANY     ANY     /login
task_list       GET         ANY     ANY     /tasks
task_done       GET         ANY     ANY     /tasks/done
task_create     GET|POST    ANY     ANY     /tasks/create
task_edit       GET|POST    ANY     ANY     /tasks/{id}/edit
task_toggle     GET|POST    ANY     ANY     /tasks/{id}/toggle
task_delete     GET|DELETE  ANY     ANY     /tasks/{id}/delete
user_list       GET         ANY     ANY     /users
user_create     GET|POST    ANY     ANY     /users/create
user_edit       GET|POST    ANY     ANY     /users/{id}/edit
user_delete     GET|DELETE  ANY     ANY     /users/{id}/delete
logout          GET         ANY     ANY     /logout
```

Améliorations des entités

Entité Tasktodo

- Ajout de la propriété « unique » à l'attribut privé $title.
- Création de l'attribut $usertodo avec une relation ManyToOne.
- Ajout de l'attribut $freshDate (date de mise à jour).
- Ajouts de plusieurs contraintes de validations : @Assert.

Entité Usertodo

- Création de l'attribut $tasktodos avec une relation OneToMany.
- Ajout de l'attribut $role (rôle utilisateur).
- Ajout de l'attribut $freshDate (date de mise à jour).
- Ajouts de plusieurs contraintes de validations : @Assert.
- Révision de la fonction « getRoles ».

```php
/**
 * @ORM\OneToMany(targetEntity="App\Entity\Tasktodo", mappedBy="usertodo",
 orphanRemoval=true, cascade={"persist", "remove"})
 */
private $tasktodos;
```

Améliorations des FormType

- Révision du FormType « UsertodoType » : appel de la classe « ChoiceType » pour sélectionner entre le ROLE_USER et le ROLE_ADMIN lors de la création ou de la modification d'un utilisateur.

- Ajouts de labels et d'attributs « placeholder » générés directement dans les champs des formulaires Twig.

```
->add('role', ChoiceType::class, [
    'required' => true,
    'label' => ' ',
    'choices' => [
        'ROLE USER' => 'ROLE_USER',
        'ROLE ADMIN' => 'ROLE_ADMIN'
    ]
])
```

2

Analyses de la qualité des codes

Outils d’analyses

- Le repository Github du projet a été relié à Codacy et à SymfonyInsight afin d’analyser en temps réel la qualité des codes modifiés ou ajoutés.

- PhpUnit a été utilisé pour vérifier le bon fonctionnement de l’application. Blackfire pour détecter certains problèmes de performances.

- Les services en ligne Jigsaw et W3C Markup Service ont également permis de détecter et de corriger certaines erreurs HTML et CSS.

Codacy

- Alertes CSS corrigées : indentation de deux espaces, suppressions des mentions « !important », séparation d'un espace entre certains caractères.

- Alertes Javascript corrigées : remplacements de guillemets simples par des guillemets doubles, révisions de variables avec le format camelCase.

- Alertes Php corrigées : révisions de variables déclarées dans les tests unitaires et fonctionnels.

Symfony Insight

Erreurs corrigées :

- Sauts de ligne (inutiles ou exigés)
- Présence de codes commentés
- Ajout d'un favicon (.ico) dans le dossier public
- Ajout d'un fichier site.webmanifest dans le dossier public
- Ajout d'un fichier robots.txt dans le dossier public
- Mise à jour de dépendances

Badge Platinium

PhpUnit

- 47 tests ont été préparés afin de s'assurer du bon fonctionnement de l'application. Ces tests unitaires et fonctionnels commandent les vérifications de 107 assertions et couvrent les fonctions majeures des fichiers des dossiers Entity, Repository, Form, Handler, Security et Controller.

- Le taux de couverture est supérieur à 70%.

```
PS C:\xampp\htdocs\todolist_project_v14> php bin/phpunit --colors
PHPUnit 8.5.14 by Sebastian Bergmann and contributors.

Testing Project Test Suite
.............................................                    47 / 47 (100%)

Time: 7.97 seconds, Memory: 68.00 MB

OK (47 tests, 107 assertions)
```

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		68.64%	243 / 354		88.41%	61 / 69		66.67%	10 / 15
Controller		99.17%	120 / 121		92.86%	13 / 14		75.00%	3 / 4
DataFixtures		0.00%	0 / 90		0.00%	0 / 2		0.00%	0 / 1
Entity		100.00%	59 / 59		100.00%	36 / 36		100.00%	2 / 2
Form		100.00%	20 / 20		100.00%	4 / 4		100.00%	2 / 2
Handler		100.00%	5 / 5		100.00%	1 / 1		100.00%	1 / 1
Repository		100.00%	4 / 4		100.00%	2 / 2		100.00%	2 / 2
Security		87.50%	35 / 40		62.50%	5 / 8		0.00%	0 / 2
Kernel.php		0.00%	0 / 15		0.00%	0 / 2		0.00%	0 / 1

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 7.0.14 using PHP 7.3.6 with Xdebug 3.0.3 and PHPUnit 8.5.14 at Tue Jun 1 22:32:07 CEST 2021.

3

Rapports de performances

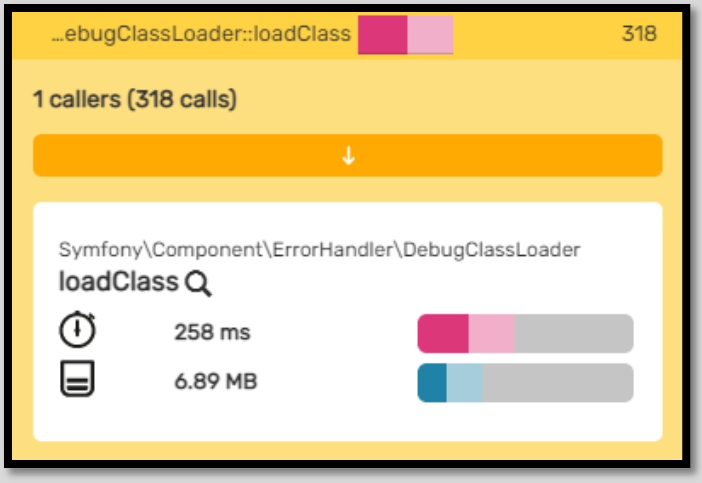
Blackfire

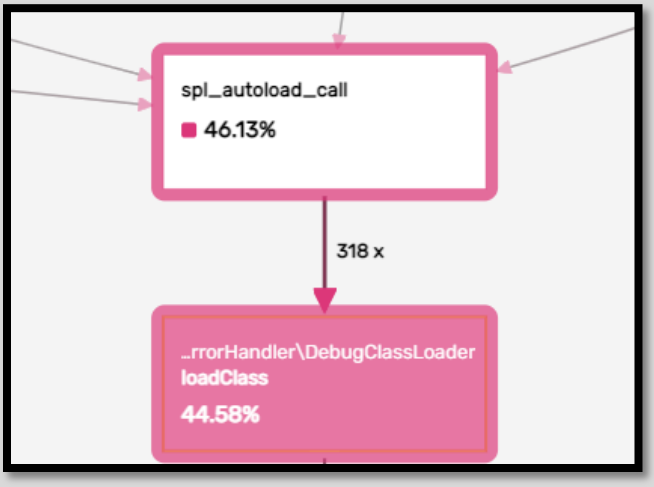
- L’outil Blackfire a été utilisé pour effectuer les principales mesures de performances de l’application.

- Le chargement des classes s’est révélé être le facteur le plus coûteux pour les temps de chargements des pages * en environnements de développement ou de production.

- La commande « *composer dump-autoload --optimize* » a été utilisée afin de mettre en œuvre un mappage de classes améliorant les temps d’initialisations des pages. Les renvois des chemins des classes se sont avérés en effet plus rapide comme le révèlent les mesures des analyses comparatives suivantes.

* Symfony\Component\ErrorHandler\DebugClassLoader::loadClass



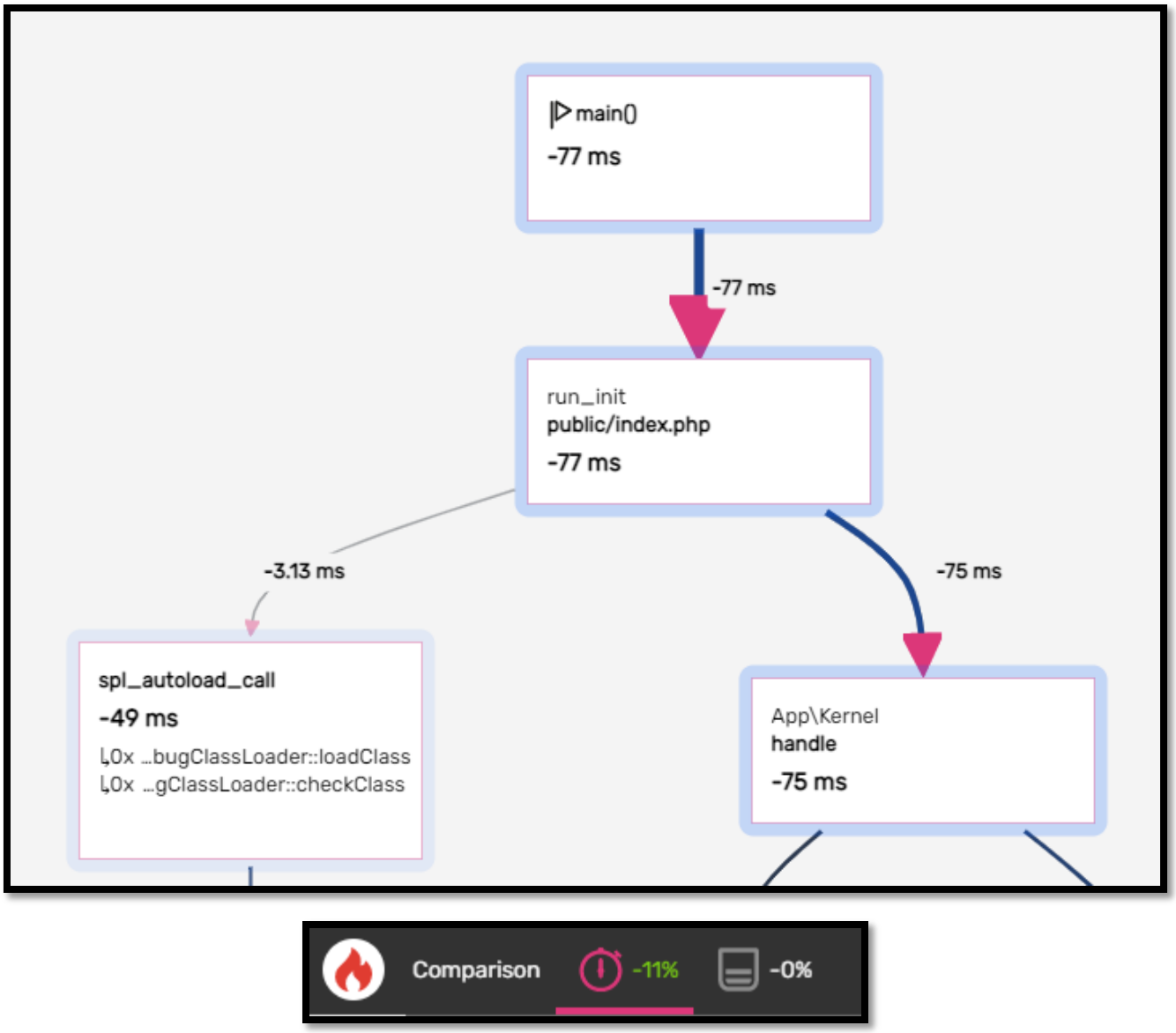


Analyses comparatives Blackfire avant et après optimisation de l'autoload de Composer

ROUTES	AVANT	APRÈS	GAINS *
/login	579 ms 23,1 MB	511 ms 23,1 MB	**- 68 ms**
/	799 ms 25,9 MB	722 ms 25,9 MB	**- 77 ms**
/tasks	890 ms 26,7 MB	740 ms 26,7 MB	**- 150 ms**
/tasks/done	846 ms 26,7 MB	742 ms 26,7 MB	**- 104 ms**
/tasks/{id}/edit	996 ms 29,8 MB	837 ms 29,8 MB	**- 159 ms**
/tasks/create	996 ms 30,1 MB	711 ms 30,1 MB	**- 285 ms**
/users	744 ms 26,4 MB	630 ms 26,4 MB	**- 113 ms**
/users/{id}/edit	962 ms 30,9 MB	761 ms 30,9 MB	**- 200 ms**
/users/create	981 ms 30,7 MB	741 ms 30,7 MB	**- 240 ms**

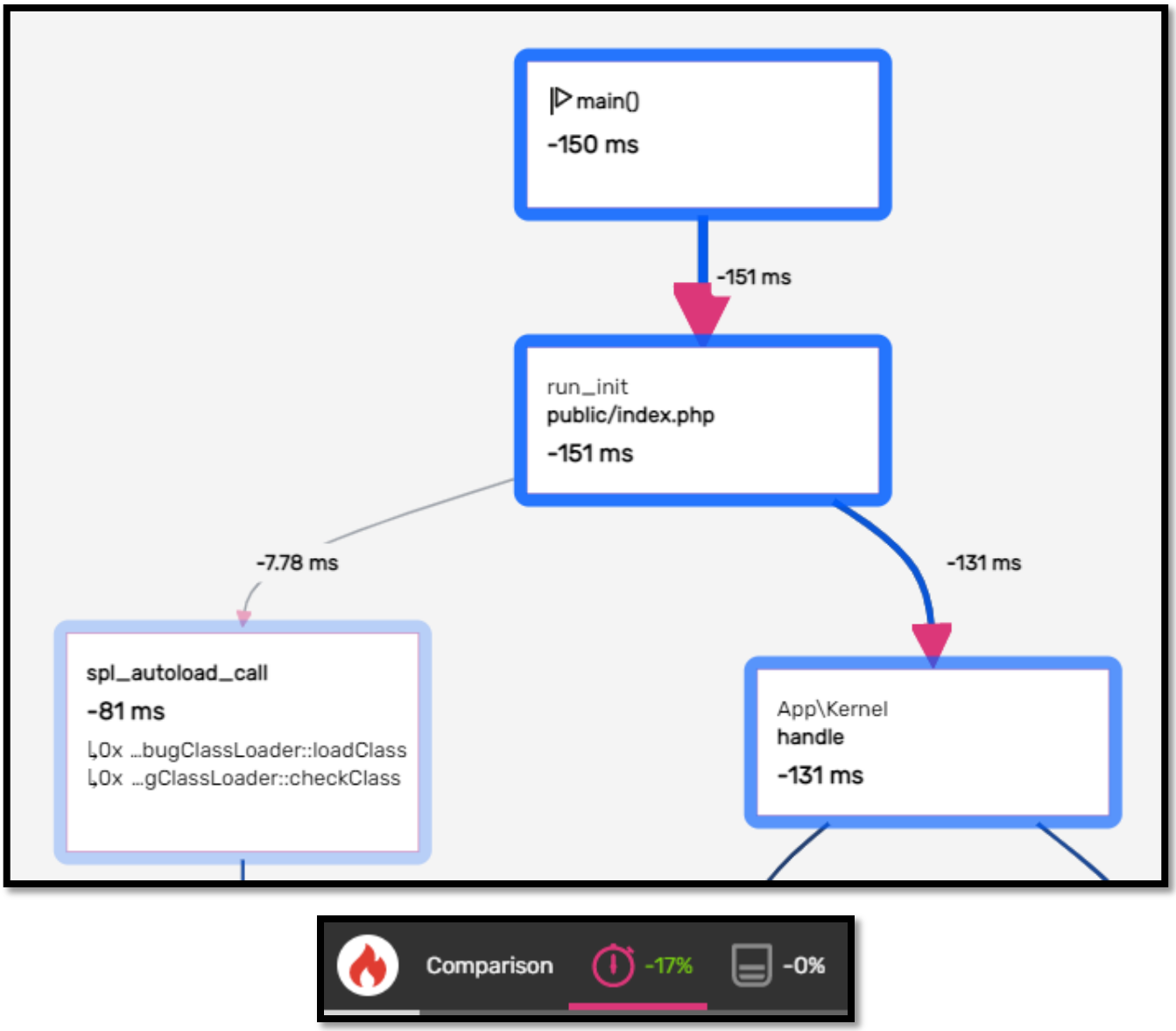
* Réductions en temps de chargement

Exemple de gain de temps de chargement sur la page d'accueil



La comparaison a permis de constater une baisse globale de 11% du temps de chargement de la page.

Exemple de gain de temps de chargement sur la page des tâches à réaliser



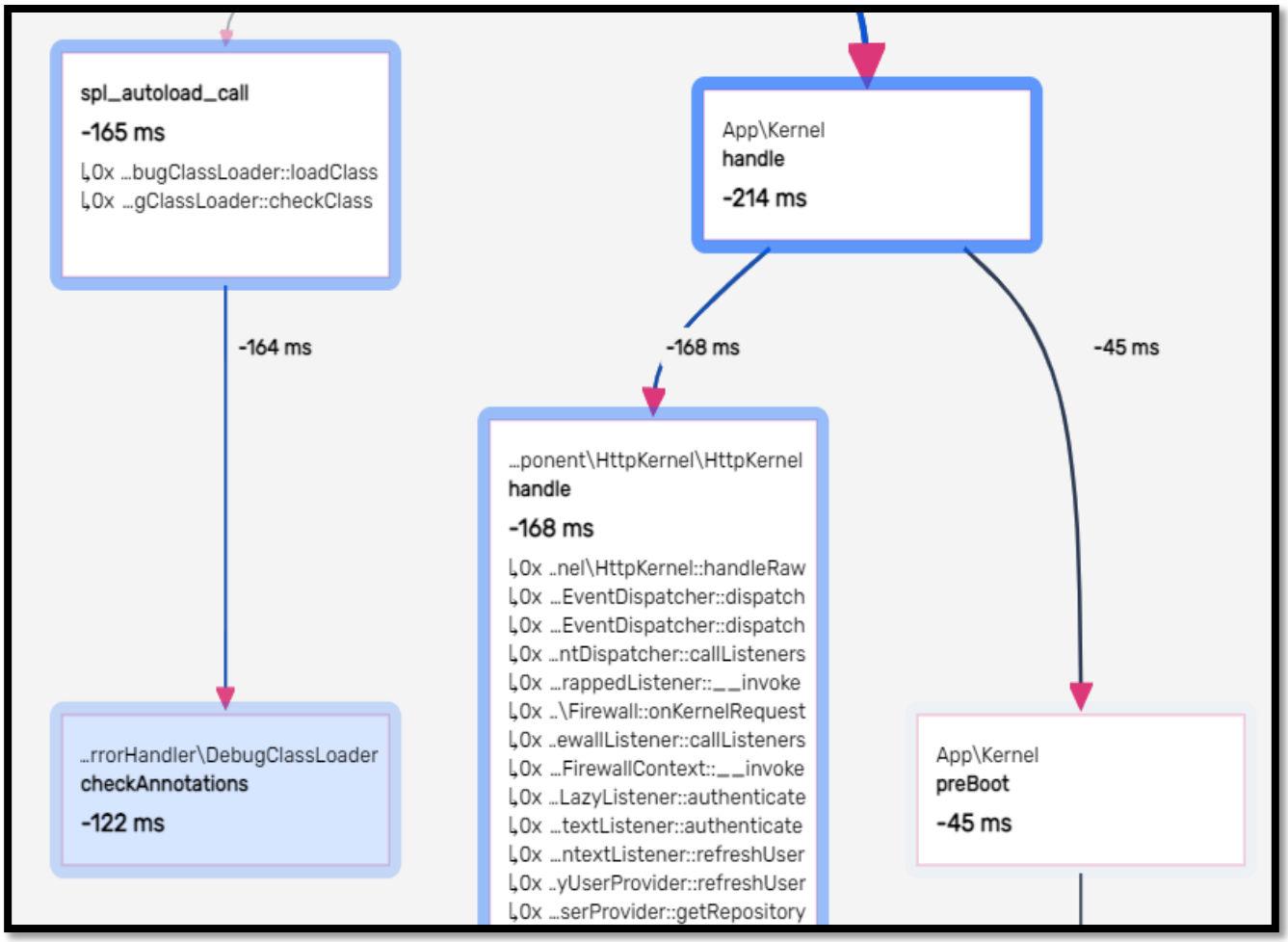
La comparaison a permis de constater une baisse de 17% du temps de chargement de la page.

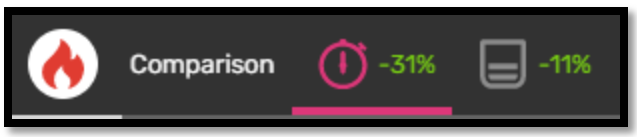
Analyses comparatives Blackfire après la désactivation de xDebug

ROUTES	AVANT	APRÈS	GAINS *
/login	511 ms 23,1 MB	470 ms 20,4 MB	**- 41 ms** **- 2,7 MB**
/	722 ms 25,9 MB	560 ms 22,9 MB	**- 162 ms** **- 3 MB**
/tasks	740 ms 26,7 MB	698 ms 23,6 MB	**- 42 ms** **- 3,1 MB**
/tasks/done	742 ms 26,7 MB	700 ms 23,6 MB	**- 42 ms** **- 3,1 MB**
/tasks/{id}/edit	837 ms 29,8 MB	810 ms 26,5 MB	**- 27 ms** **- 3,3 MB**
/tasks/create	711 ms 30,1 MB	647 ms 26,3 MB	**- 64 ms** **- 3,8 MB**
/users	630 ms 26,4 MB	615 ms 23,3 MB	**- 15 ms** **- 3,1 MB**
/users/{id}/edit	761 ms 30,9 MB	706 ms 27,3 MB	**- 55 ms** **- 3,6 MB**
/users/create	741 ms 30,7 MB	691 ms 27,1 MB	**- 50 ms** **- 3,6 MB**

* Réductions en temps de chargement et en mémoire utilisée

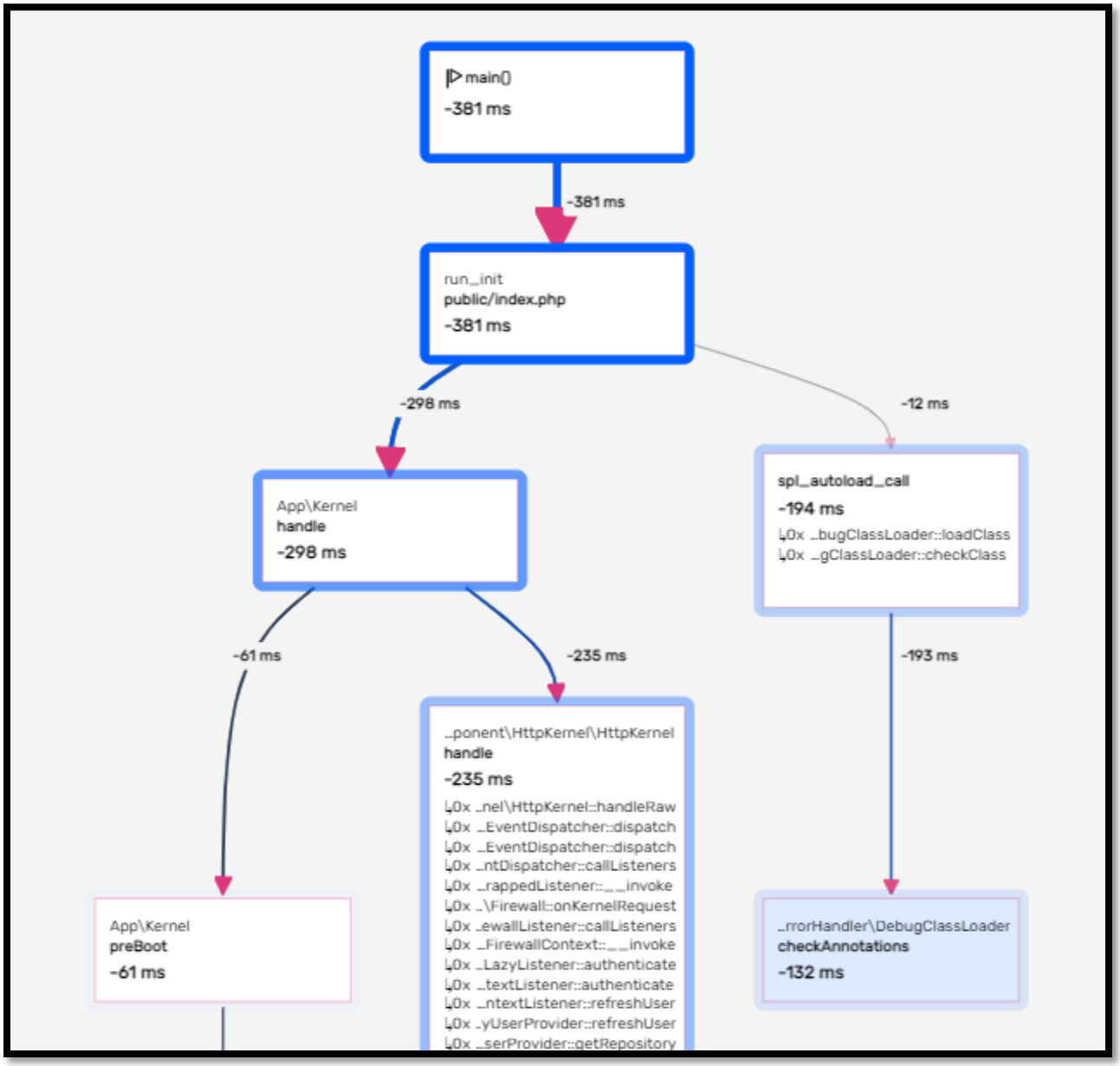
Exemple de gains de performances sur la page de gestion des utilisateurs





La comparaison a permis de constater une baisse de 31% du temps de chargement de la page et une réduction de 11% de la mémoire utilisée

Exemple de gains de performances sur la page des tâches terminées



Baisse de 35% du temps de chargement et réduction de 11% de la mémoire utilisée

Relevés des temps de chargement par pages selon le profiler de Symfony

ROUTES	Temps *
/login	336 ms
/	394 ms
/tasks	480 ms
/tasks/done	462 ms
/tasks/{id}/edit	566 ms
/tasks/create	509 ms
/users	422 ms
/users/{id}/edit	514 ms
/users/create	518 ms

* En millisecondes

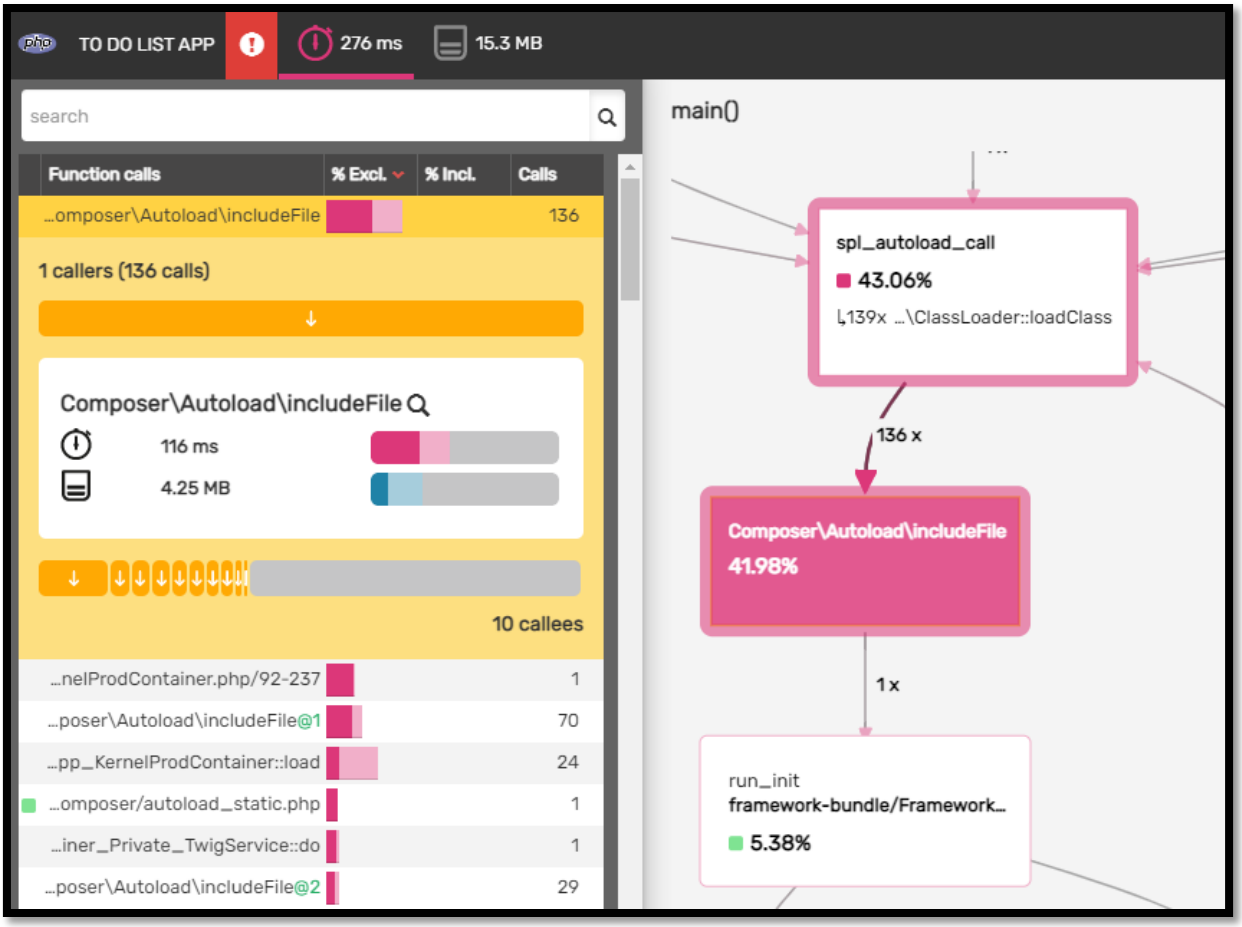
Mesures optimales en environnement de production

ROUTES	MESURES *	GAINS **
/login	207 ms 12,4 MB	**- 263 ms** **- 8 MB**
/	276 ms 15,3 MB	**- 284 ms** **- 7,6 MB**
/tasks	258 ms 15,4 MB	**- 440 ms** **- 8,2 MB**
/tasks/done	252 ms 15,4 MB	**- 448 ms** **- 8,2 MB**
/tasks/{id}/edit	299 ms 18,1 MB	**- 511 ms** **- 8,4 MB**
/tasks/create	308 ms 18 MB	**- 339 ms** **- 8,3 MB**
/users	250 ms 15,4 MB	**- 365 ms** **- 7,9 MB**
/users/{id}/edit	307 ms 18,6 MB	**- 399 ms** **- 8,7 MB**
/users/create	310 ms 18,5 MB	**- 381 ms** **- 8,6 MB**

* Temps de chargement et mémoire utilisée

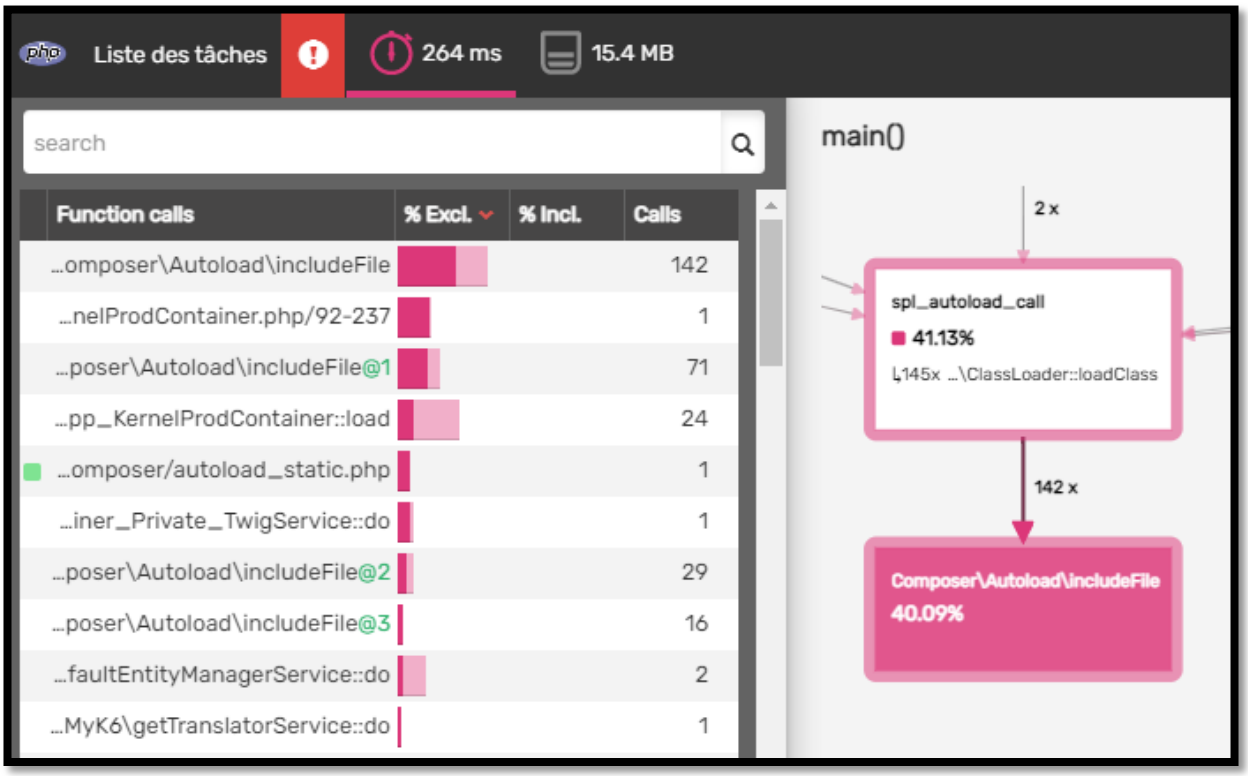
** Par rapport aux relevés effectués en mode développement

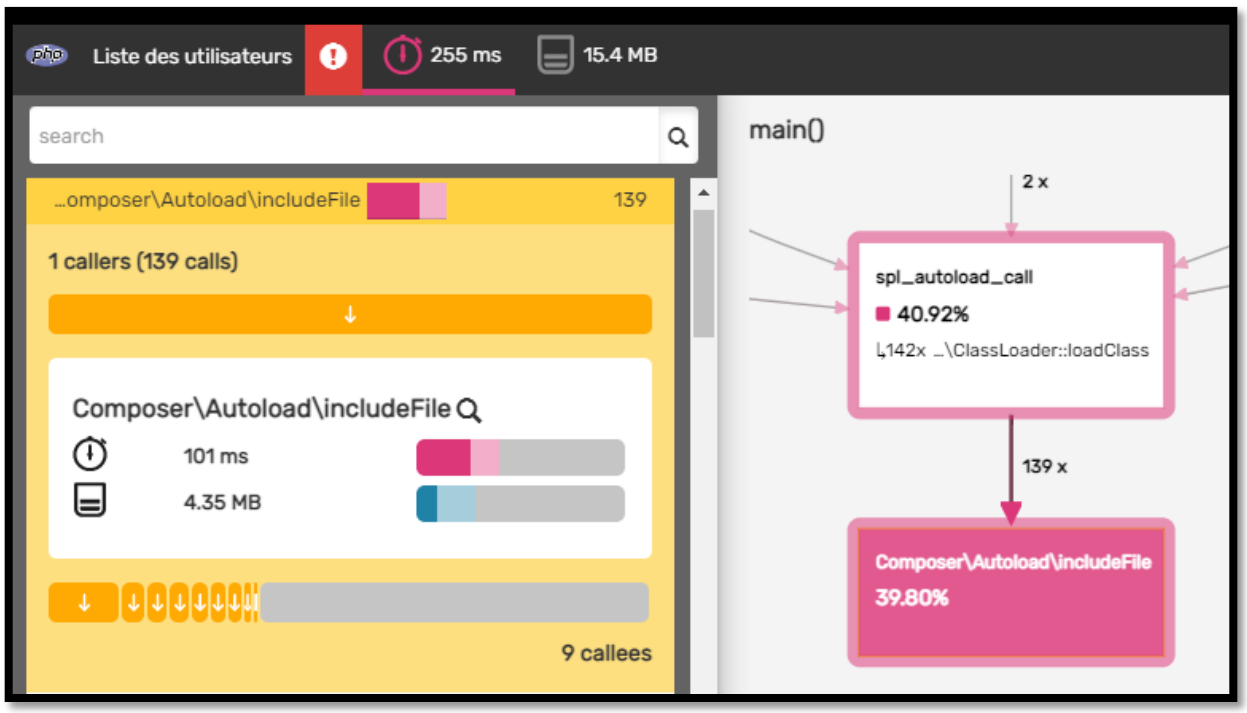
Exemple de métriques sur la page d'accueil en environnement de production



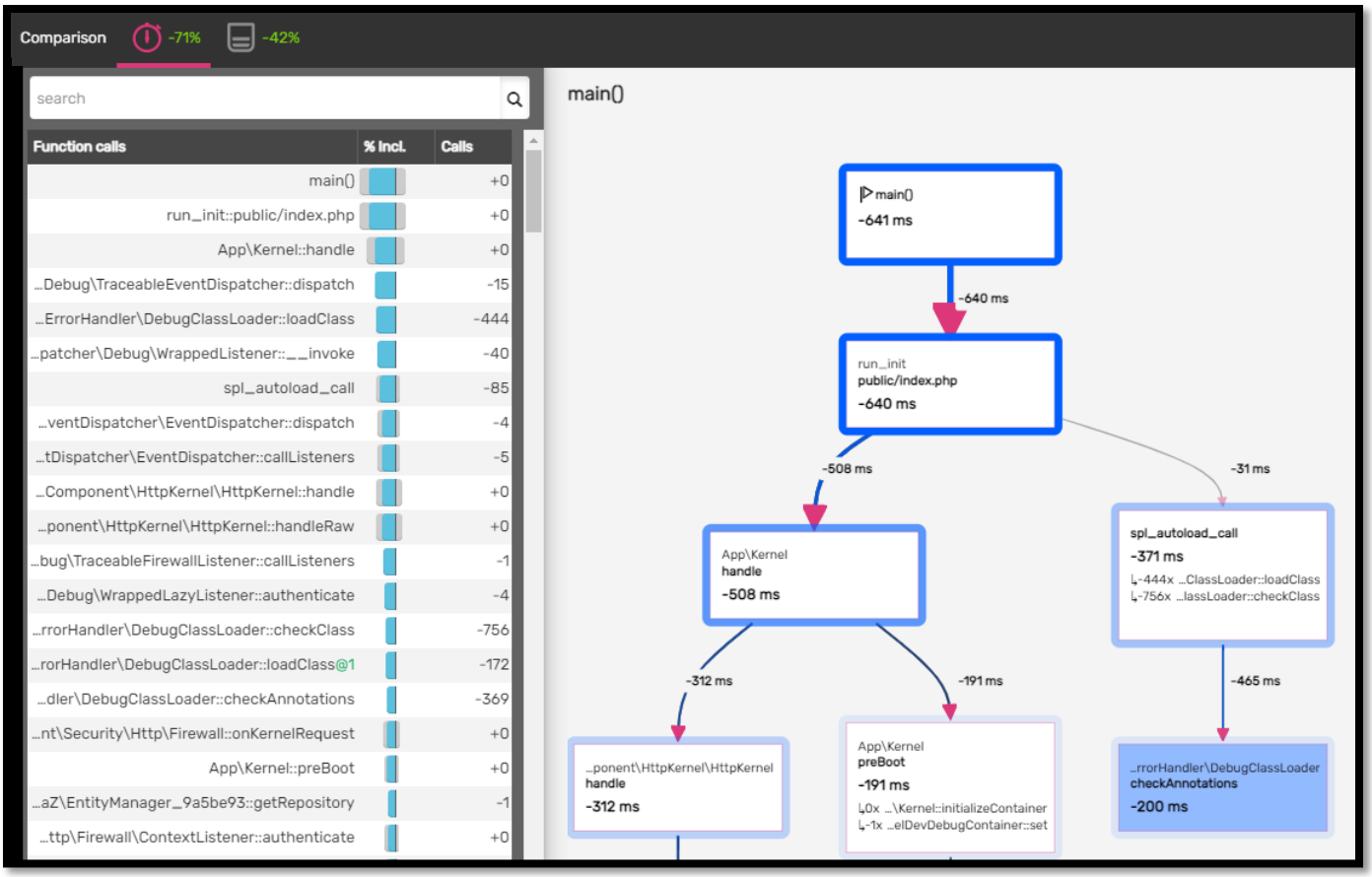
L'autoload des fichiers reste le principal facteur affectant les performances de l'application, après son optimisation en environnement de production.

Mesures sur la page des tâches et la page de gestion des utilisateurs



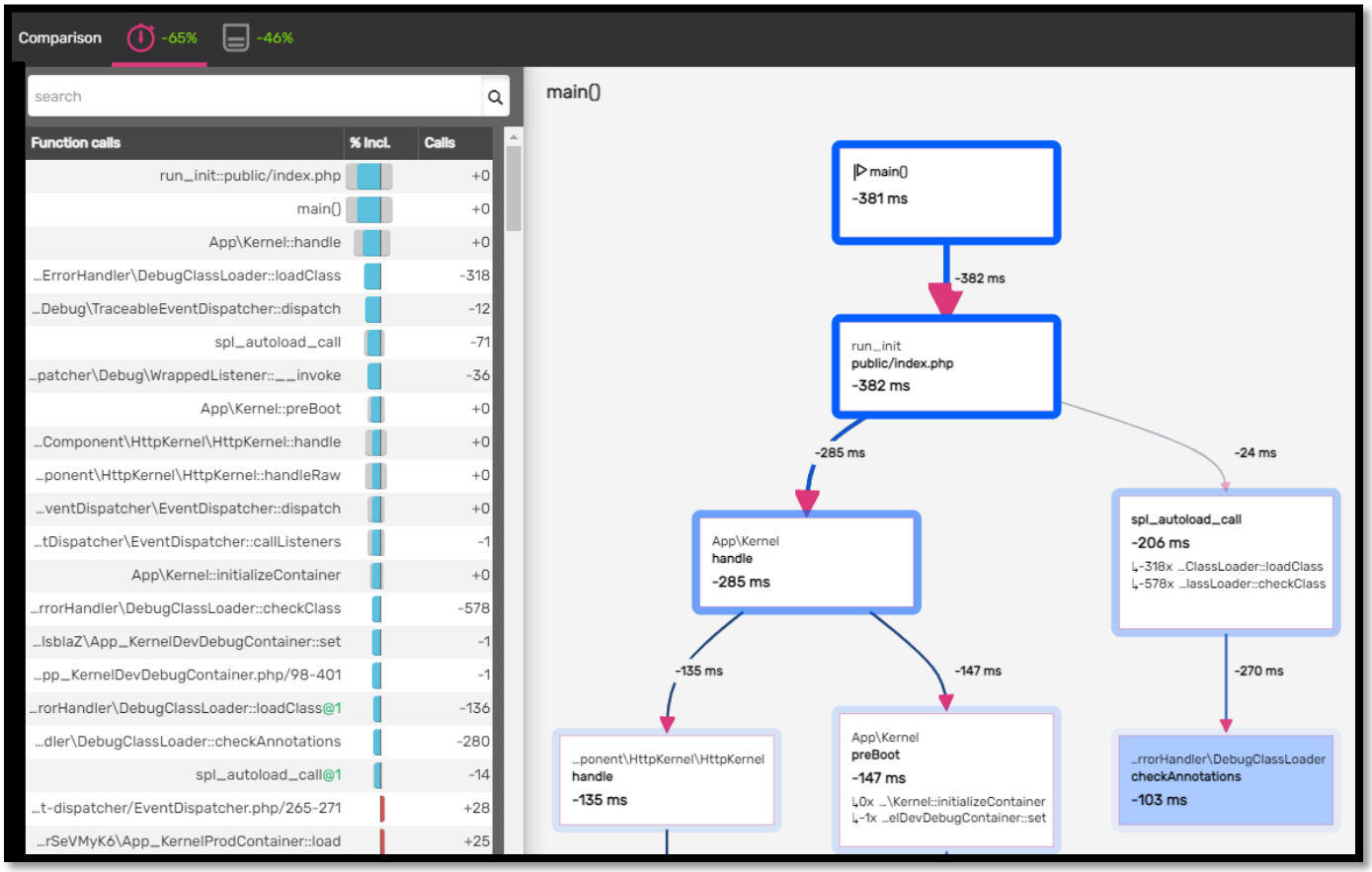


Analyse comparative des gains en performances sur la page de gestion des utilisateurs



Le passage en environnement de production a permis de constater une baisse de 71% du temps de chargement de la page et une réduction de 42% de la mémoire utilisée.

Analyse comparative des gains en performances sur la page de connexion



Le passage en environnement de production a permis de constater une baisse de 65% du temps de chargement de la page et une réduction de 46% de la mémoire utilisée.

Autres solutions apportées pour les gains de performances

- Mise en place d'une pagination sur les listes des tâches et sur la page de gestion des utilisateurs afin de limiter les temps de chargements (10 par page). Limitation du nombre maximal d'items affichables par page (max 100).

- Installation de Webpack Encore et téléchargements des dépendances via NPM (Bootstrap, Jquery, Glyphicons). La construction de versions minifiées des fichiers Javascript et CSS dans le dossier « /build » a un impact positif en terme de performance par rapport aux simples déclarations des assets dans le projet de départ.

- Ajout d'une section Cookie « remember_me » dans le fichier security.yaml et paramétrage du Cookie avec un temps de validité équivalent à une semaine. Les temps de chargement liés aux étapes d'authentification se trouvent considérablement réduits pour l'usage global de l'application.

- Ajout du paramètre « lazy: true » dans la section « firewalls » du fichier security.yaml afin de mettre en cache les urls ne nécessitant pas d'authentification.

Perspectives d'améliorations supplémentaires des performances de l'application

- Implémentation de sections de mise en cache pour certains fragments des templates Twig : {% cache %}{% endcache %}. Opération qui nécessite au préalable l'installation de la dépendance Twig cache-extra (« composer require twig/cache-extra »).

- Ajouts d'annotations @cache aux fonctions des controllers pour définir des paramètres spéciaux de mise en cache selon les spécificités des routes de l'application (modifications des en-têtes, précisions sur le temps d'expiration ou de mise à jour, etags, max-age…). L'annotation @cache peut également être définie sur l'intégralité d'une classe pour affecter toutes ses fonctions.

- En plus de l'optimisation de l'autoload de Composer, la commande « *composer dump-autoload --no-dev --classmap-authoritative* » permet en environnement de production de mieux préciser encore le mappage de classes et d'éviter d'autres analyses relativement coûteuses en temps de chargements.

Perspectives d'améliorations en environnement de production

- Plusieurs tests effectués en environnement de production ont révélé parfois certaines lenteurs de chargements dûs principalement aux temps de réponses du serveur. Pour obtenir des performances optimales, le choix d'un système de base de données rapide est conseillé.

- L'adoption d'une mise en cache plus longue pourrait aussi être un atout supplémentaire, en précisant notamment dans les en-têtes les délais d'expiration : la conservation de ressources statiques diminuant les charges sur le serveur.

Analyses Dareboost et PageSpeed Insights