****

**Introduction :**

**Attendu :**

Les fondateurs souhaitent pérenniser le développement de l’application. Cela dit, ils souhaitent dans un premier temps faire un état des lieux de la dette technique de l’application.

Au terme de votre travail effectué sur l’application, il vous est demandé de produire un audit de code sur les deux axes suivants : la qualité de code et la performance.

**Partie : Présentation**

**Partie : Audit de qualité du code**

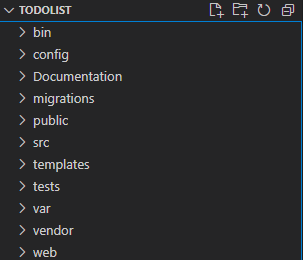
**Partie : Performance de l’application**

**Partie : Tests automatisés**

**Partie : Comment contribuer**

PRESENTATION

* **Architecture.**



**Bin: Contient les fichiers de commandes permettant d’effectuer des actions sur un projet Symfony.**

**Config : Configuration des packages, services et routes (YAML)**

**Migration : Contient les fichier de migrations Doctrine -> BDD**

**Public : Point d’entrée de l’application, index.php. Contient les images.**

**SRC : Cœur du projet ! Dossier qui contient la logique de votre application.**

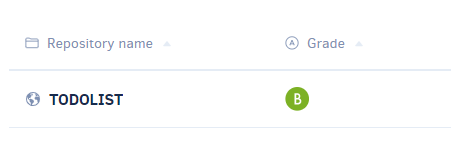
**Templates : Contient nos Views. Symfony utilise le moteur de Template Twig par défaut.**

**Var : Cache et fichiers de log.**

**Vendor : Packages de Symfony listés dans le Fichier Composer.json**

AUDIT DE QUALITE DU CODE

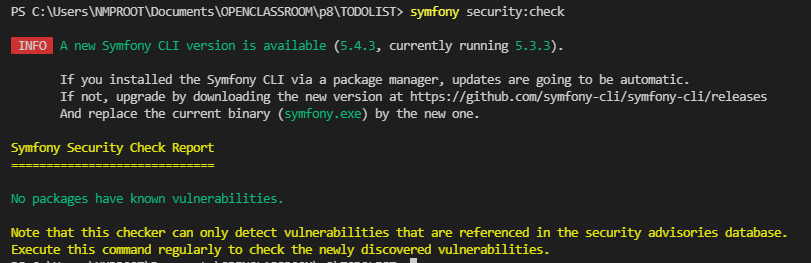
* Analyse du code.



Le projet n’obtient que la note de B sous codacy.

# Sécurité.

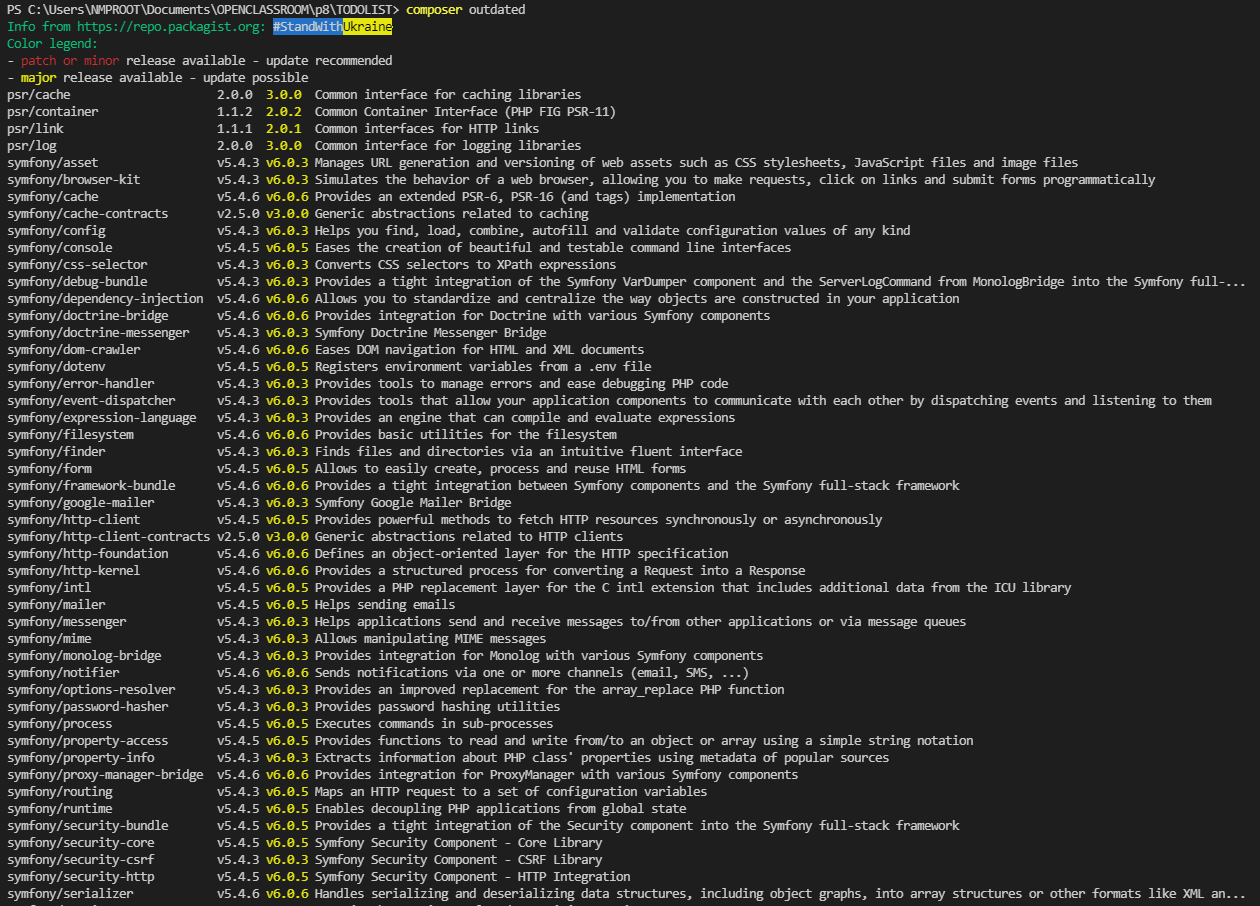
Commande : *symfony security:check*



Pas de vulnérabilité.

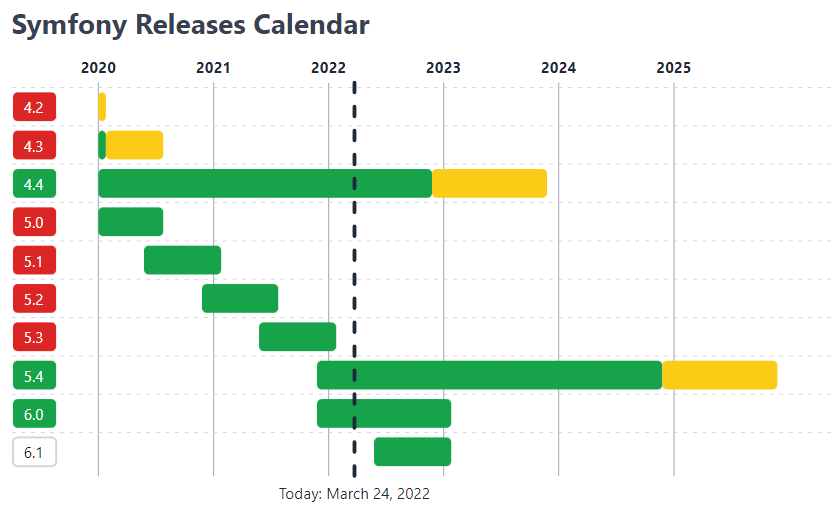
* **Mise à jour.**

Commande : *composer outdated*

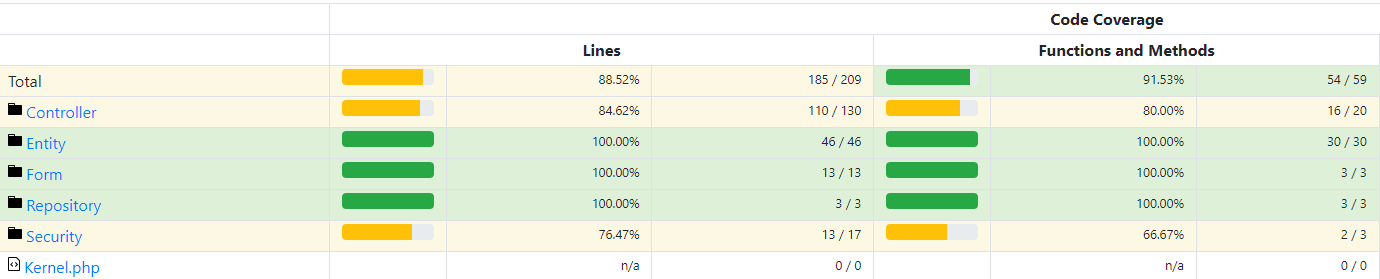


Mise à jour possible vers symfony 6.

Cependant nous resterons sur la versions 5.4 qui est la LTS (Long-Term Support Release).



* Test unitaire
* public function testCreateAction()
* {
* // Login grâce a la fonction loginuser()
* $userRepository = static::getContainer()->get(UserRepository::class);
* $userTest = $userRepository->findOneByEmail('test@test.fr');
* $this->client->loginUser($userTest);
* // Nous accédons à la page "mes taches" de l'user ayant l'ID 1
* $crawler = $this->client->request('GET', '/tasks/1');
* $this->assertEquals(200, $this->client->getResponse()->getStatusCode());
* //Recherche du bouton "ajouter"
* $buttonCrawlerNode = $crawler->selectButton("Ajouter");
* // Sélection du formulaire associé a ce bouton
* $form = $buttonCrawlerNode->form();
* // On remplit le formulaire
* $form['task[title]'] = 'PHPUNIT';
* $form['task[content]'] = 'Faire les test unitaires';
* //On soumet le formulaire
* $this->client->submit($form);
* // Vérification de la redirection et du code retour 200
* $this->assertTrue($this->client->getResponse()->isRedirect());
* $crawler = $this->client->followRedirect();
* $this->assertEquals(200, $this->client->getResponse()->getStatusCode());
* }

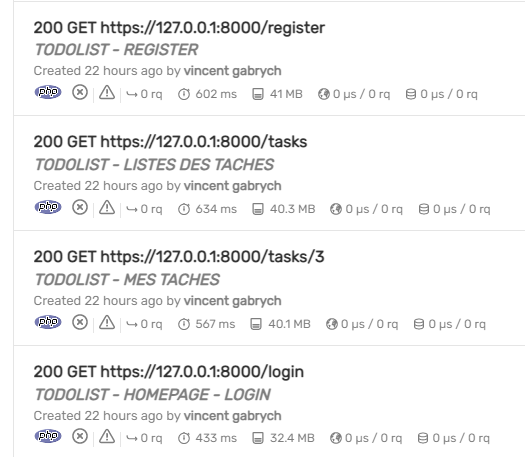


**PERFORMANCE DE L’APPLICATION**

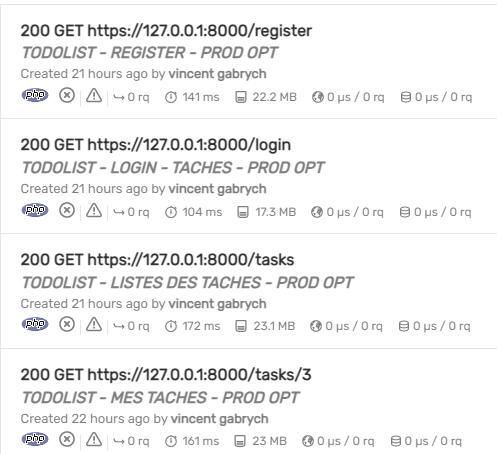
* Tests de performance

Les tests de performance sont réalisés avec Blackfire.

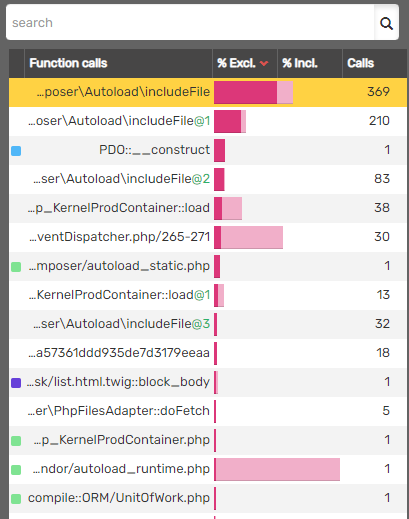
En Dev avec profiler :

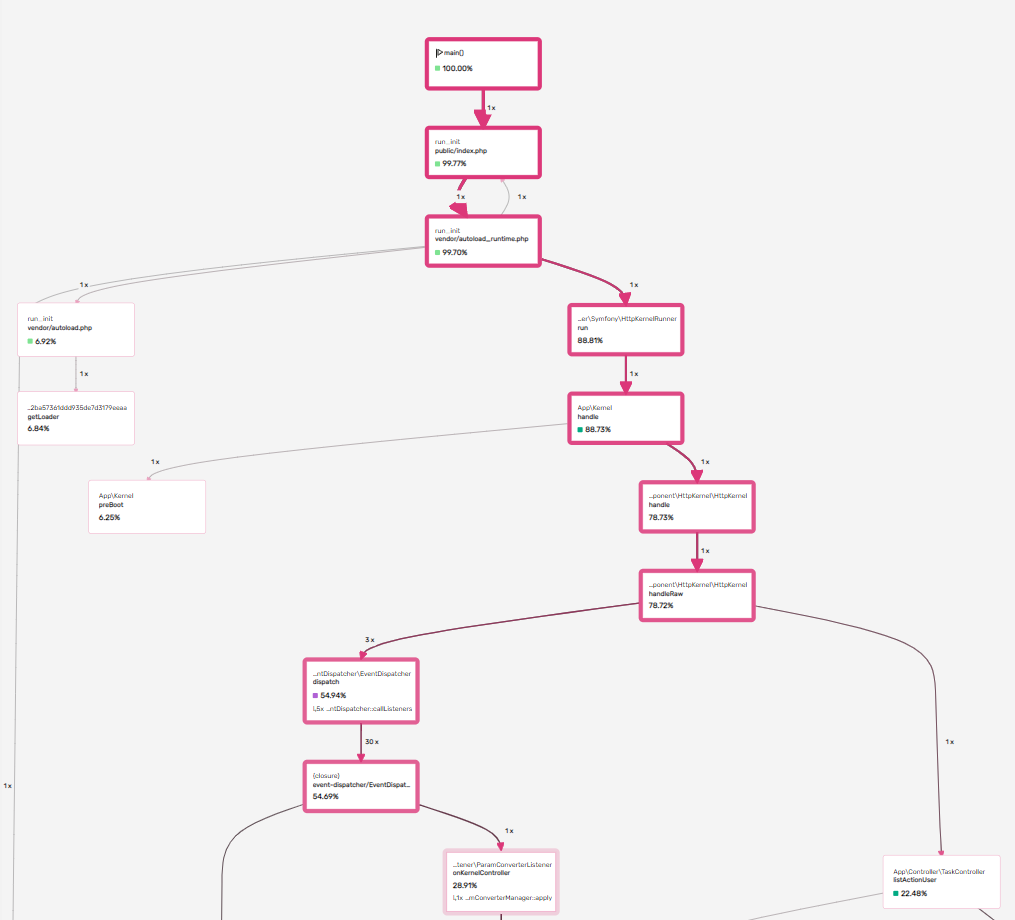


Passage en Prod :



Analyse :





**CONTRIBUER**