****

**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**

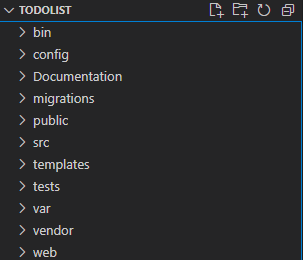
**Partie : Performance de l’application**

**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**

* **Paquet :**

symfony/asset : Manages URL generation and versioning of web assets such as CSS stylesheets, JavaScript files and image files

symfony/browser-kit : Simulates the behavior of a web browser, allowing you to make requests, click on links and submit forms programmatically

symfony/debug-bundle : Provides a tight integration of the Symfony VarDumper component and the ServerLogCommand from MonologBridge into the Symfony full-...

symfony/dependency-injection : Allows you to standardize and centralize the way objects are constructed in your application

symfony/doctrine-bridge : Provides integration for Doctrine with various Symfony components

symfony/dom-crawler : Eases DOM navigation for HTML and XML documents

symfony/dotenv : Registers environment variables from a .env file

symfony/form : Allows to easily create, process and reuse HTML forms

symfony/framework-bundle :Provides a tight integration between Symfony components and the Symfony full-stack framework

symfony/google-mailer : Symfony Google Mailer Bridge

symfony/ : Helps sending emails

symfony/messenger : Helps applications send and receive messages to/from other applications or via message queues

symfony/monolog-bridge : Provides integration for Monolog with various Symfony components

symfony/options-resolver : Provides an improved replacement for the array\_replace PHP function

symfony/password-hasher : Provides password hashing utilities

symfony/security-bundle : Provides a tight integration of the Security component into the Symfony full-stack framework

symfony/twig-bridge : Provides integration for Twig with various Symfony components

symfony/twig-bundle : Provides a tight integration of Twig into the Symfony full-stack framework

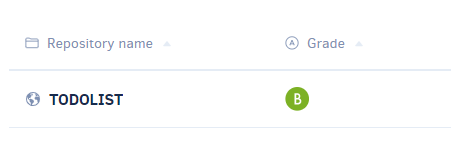
symfony/uid : Provides an object-oriented API to generate and represent UIDs

symfony/web-profiler-bundle : Provides a development tool that gives detailed information about the execution of any request

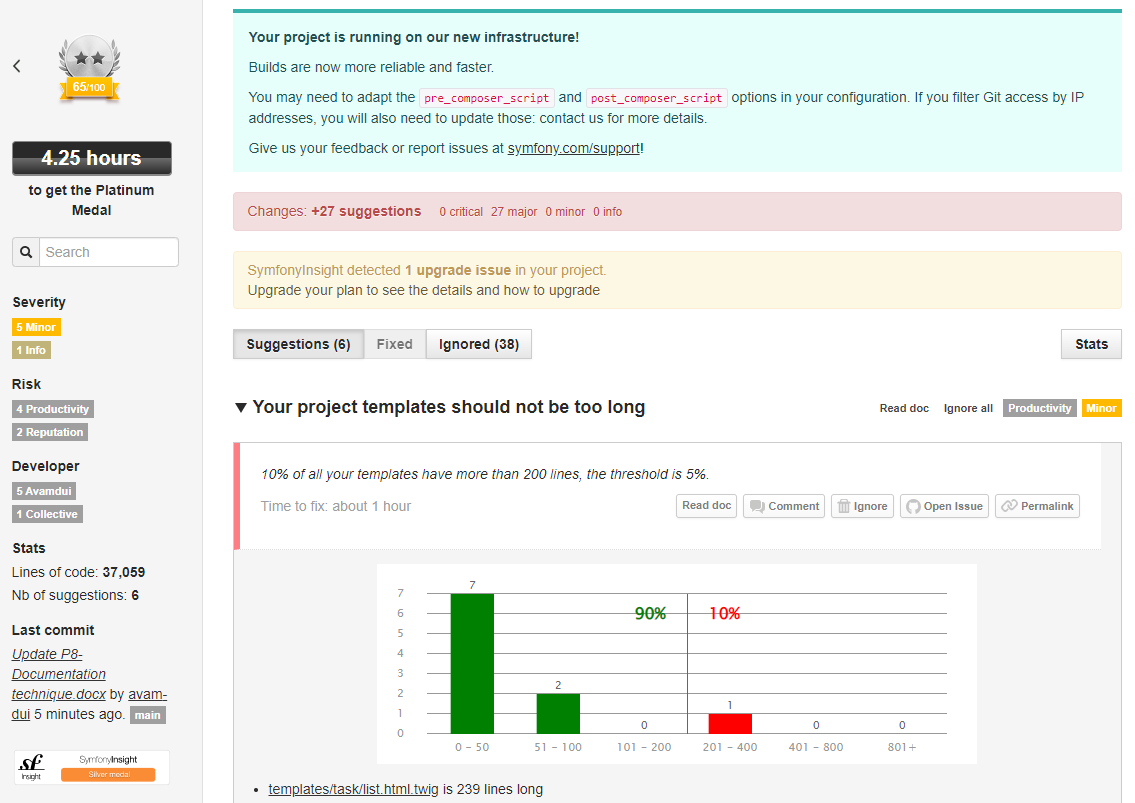
symfony/yaml : Loads and dumps YAML files

AUDIT DE QUALITE DU CODE

* Analyse du code.



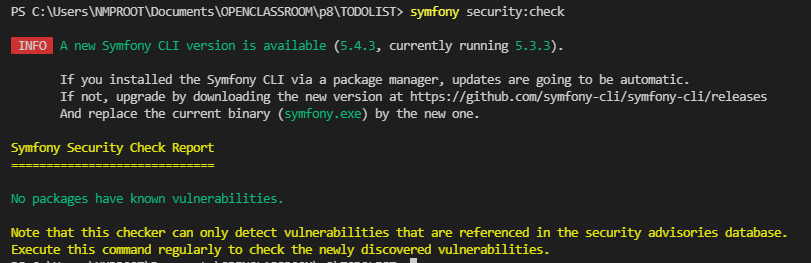
Le projet n’obtient que la note de B sous codacy.



Le projet obtient la médaille d’argent sous symfony insight

# Sécurité.

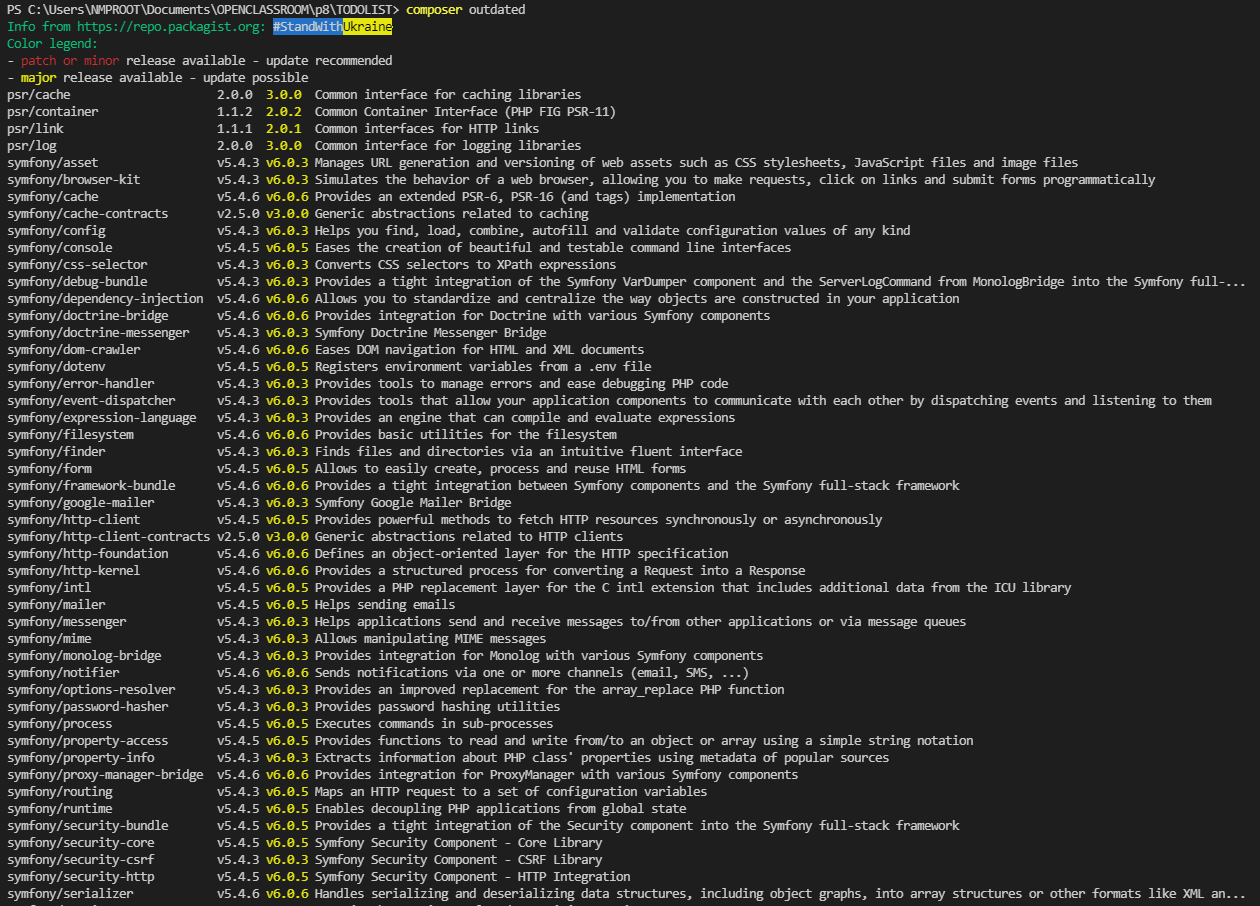
Commande : *symfony security:check*



Pas de vulnérabilité.

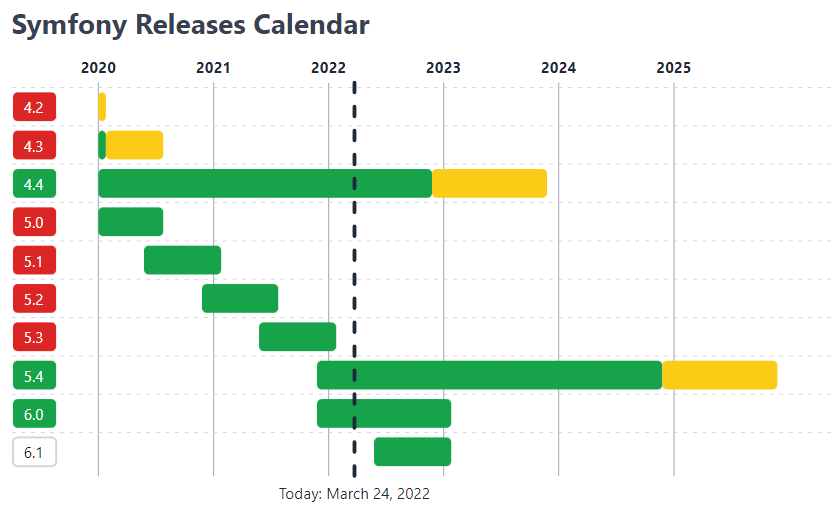
* **Mise à jour.**

Commande : *composer outdated*

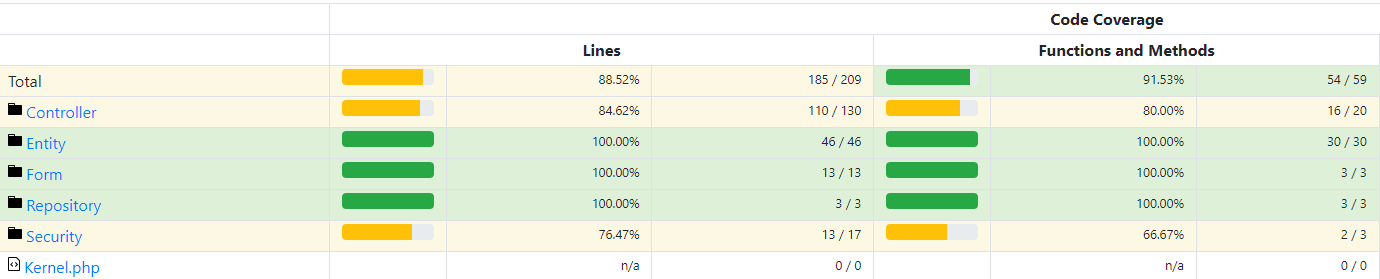


Mise à jour possible vers symfony 6.

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());
* }

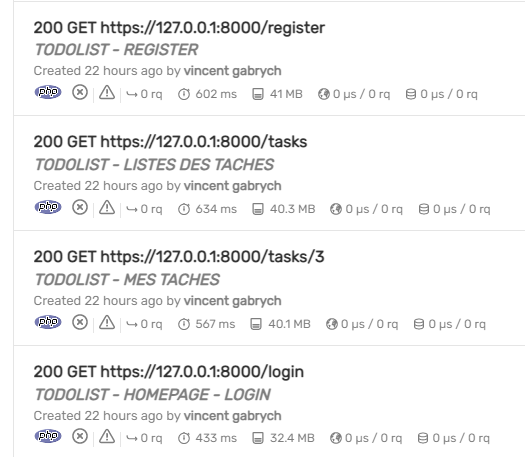


**PERFORMANCE DE L’APPLICATION**

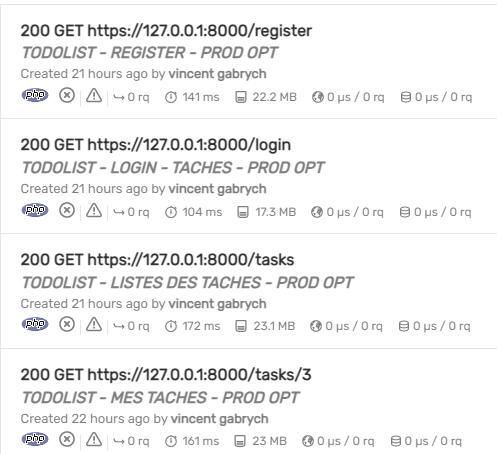
* Tests de performance

Les tests de performance sont réalisés avec Blackfire.

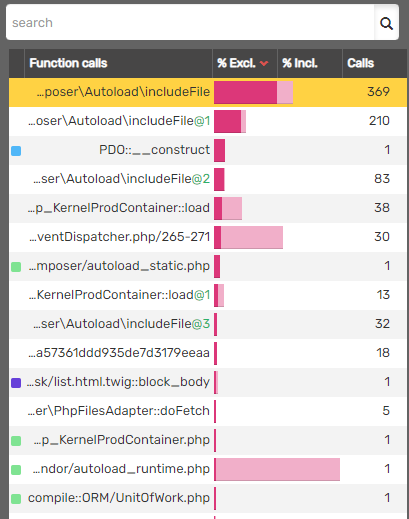
En Dev avec profiler :

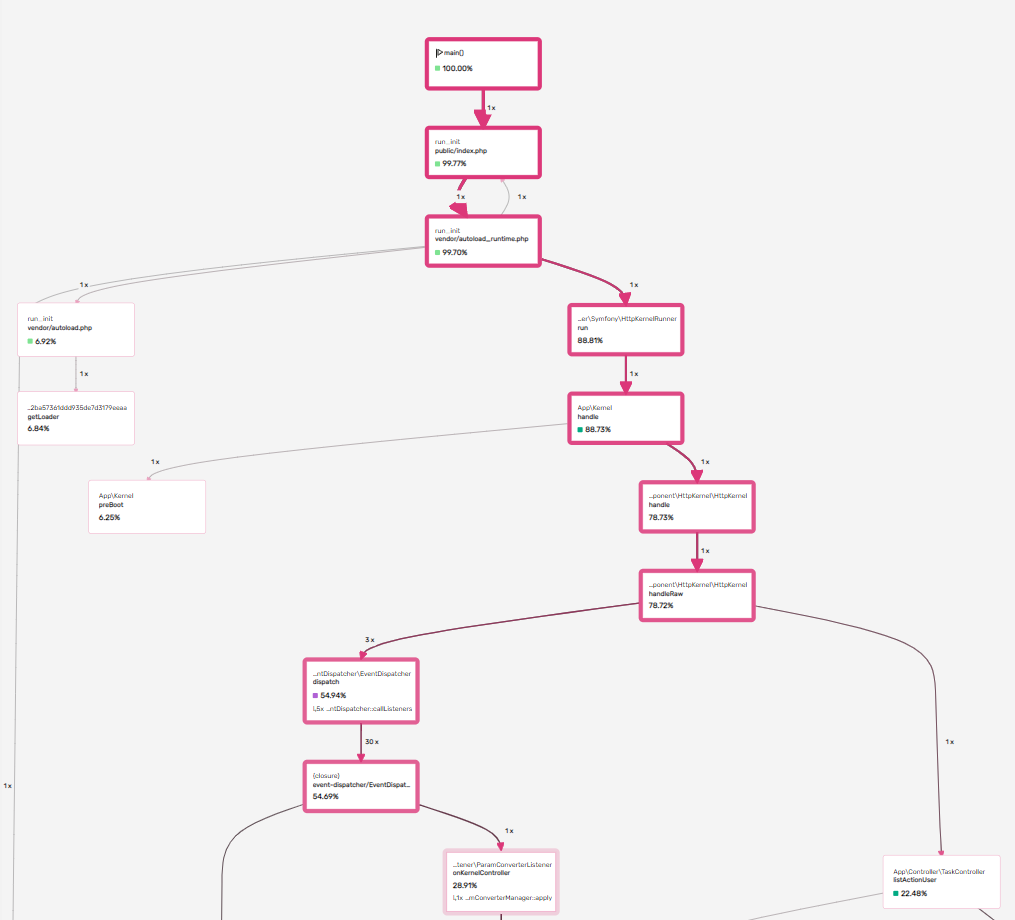


Passage en Prod :



Analyse :





**CONTRIBUER**

**# TODOLIST**

**# Avant propos**

- Le projet fonctionne sur PHP 8.0.15

- Le projet est basé sur le framework symfony 5.4.4 (Doctrine, Twig et PhpUnit)

- Git du projet : `git clone https://github.com/avamdui/TODOLIST`

**# Comment Contribuer au projet**

1. Cloner et Installer le repository sur votre serveur (voir le README.md)

  -  Modifier le .env avec vos informations.

  -  Installez les dependances : composer install

  -  Mettre en place la BDD :

      php bin/console doctrine:database:create

       php bin/console doctrine:migrations:migrate

2. Créez une branche à partir de *\*master\** : git checkout -b nom de la branche

3. Ecrivez un Issue sur les modifications que vous allez apporter

4. Ecrivez votre code EN RESPECTANT LES BONNES PRATIQUES

5. Ecrivez des Commit Clairs et precis avant de faire un Push de la branche : git push origin maBranche

6. Mettez a jour vos issues

7. Faites un *\*Pull Request\** et attendez sa validation

**# Les bonnes pratiques**

**#  1. le code**

    Vous devez respecter :

    - Le PSR 2 au minimum

    - Les standards du code de Symfony (`https://symfony.com/doc/current/contributing/code/standards.html`)

    - Les conventions du code de Symfony (`https://symfony.com/doc/5.2/contributing/code/conventions.html`)

**# 2. les bundles**

    - Toute installation de bundle PHP doit se faire avec "Composer OBLIGATOIREMENT"

**# 3. Git**

    Vous devez faire les choses dans cet ordre :

    - Nouvelle branche à partir de master duement nomée

    - Commit Correctement commentés

    - Issue Correctement commentées et documentées

    - pull Request OBLIGATOIRE

    - Seul le chef de projet peu faire un "merge" sur "master" après révision de votre code.

    - Faire un update sur le code principal : git pull origin master

**# 4) Tests unitaires et fonctionnels**

    - Toute nouvelle fonctionnalité doit avoir des tests associés

    - Vous devez respecter un taux de couverture au delà de 70%

    - PhpUnit est à votre disposition pour créer vos tests

       \* Ecrivez vos tests dans le dossier /test

       \* Utiliser MakerBundle's make:test pour creer une squelette de test!

       \* Lancer les tests avec la commande vendor/bin/phpunit --coverage-html web/test-coverage pour générer un rapport de couverture

**# 5) Architecture de fichier**

    - Respectez l'architecture de symfony 5 pour vos fichiers PHP ( src\Controller\... )

    - Les vues devront être dans le repertoire templates.