UE : GENIE LOGICIEL

Enseignant : **Dr HAYATOU OUMAROU**

Année académique 2020/2021

|  |  |  |
| --- | --- | --- |
| **#** | **Noms et Prénoms** | **Matricule** |
| 1 | NGABOU TCHINDA Gabrelle |  |
| 2 | NGOUNOU Youssouf Sokamte |  |
| 3 | NGUELEODAI Reved Maina |  |
| 4 | NJOYA Youchaou |  |
| 5 | NZOULEWA TEWOUNDOUO Dorine | 16A0240FS |

Par les étudiants (Groupe 7)

**Mesure de la qualité du code source d’un projet**

Dans le cadre de notre TPE, il nous a été demandé d’analyser le code d’un projet de notre choix à l’aide des outils d’analyse de notre choix.

Pour un travail collaboratif et à distance, nous avons déposé le projet à analyser dans un dépôt **github** à l’adresse : <https://github.com/sheriffnjoya-dev/TPE-GL-Gpe7.git>

C’est un petit projet web pour les gestions des étudiants, basés sur PHP, HTML et CSS.

Nous avons utilisé les outils suivants pour analyser et mesurer la qualité du code source de notre projet :

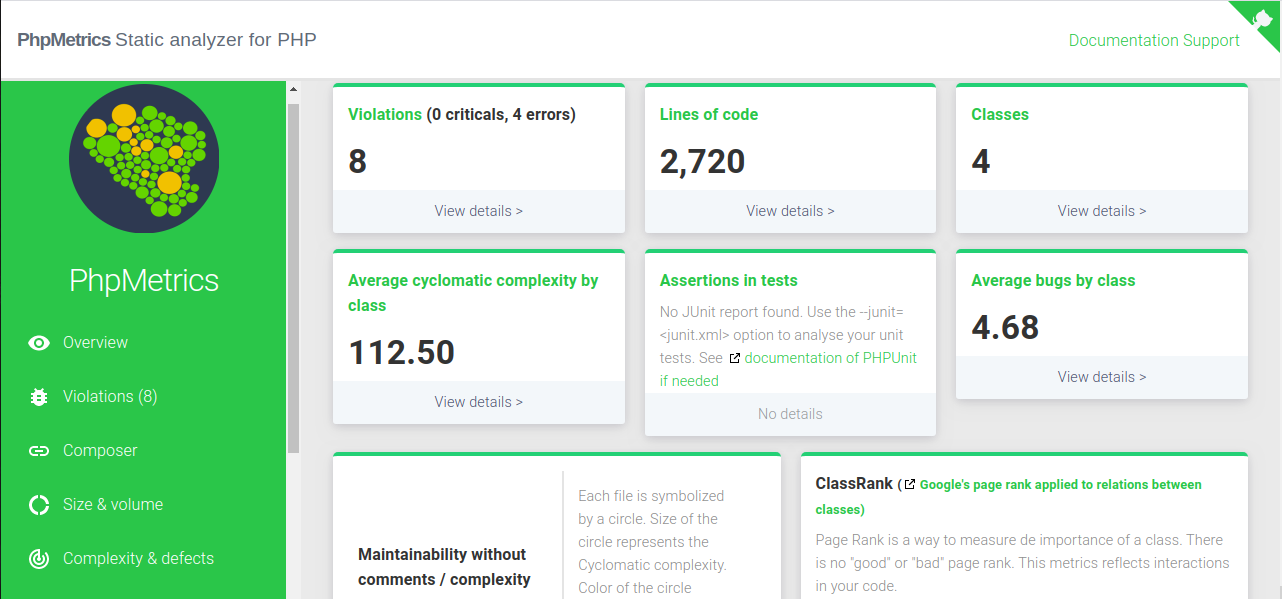
* **PHP Metrics**
* **PHPCpd**
* **CLOC**
* **PHPINSIGHTS**
* **SLOCCOUNT**
* **Sonar**

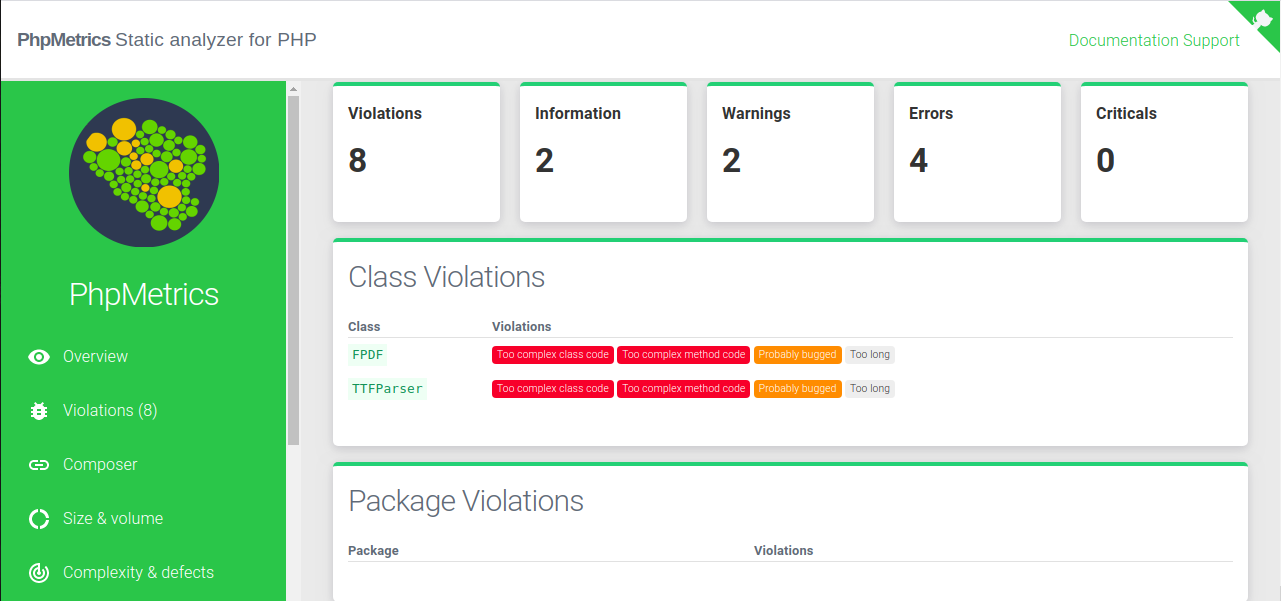
# Résultat d’analyse avec l’outil PHP Metrics

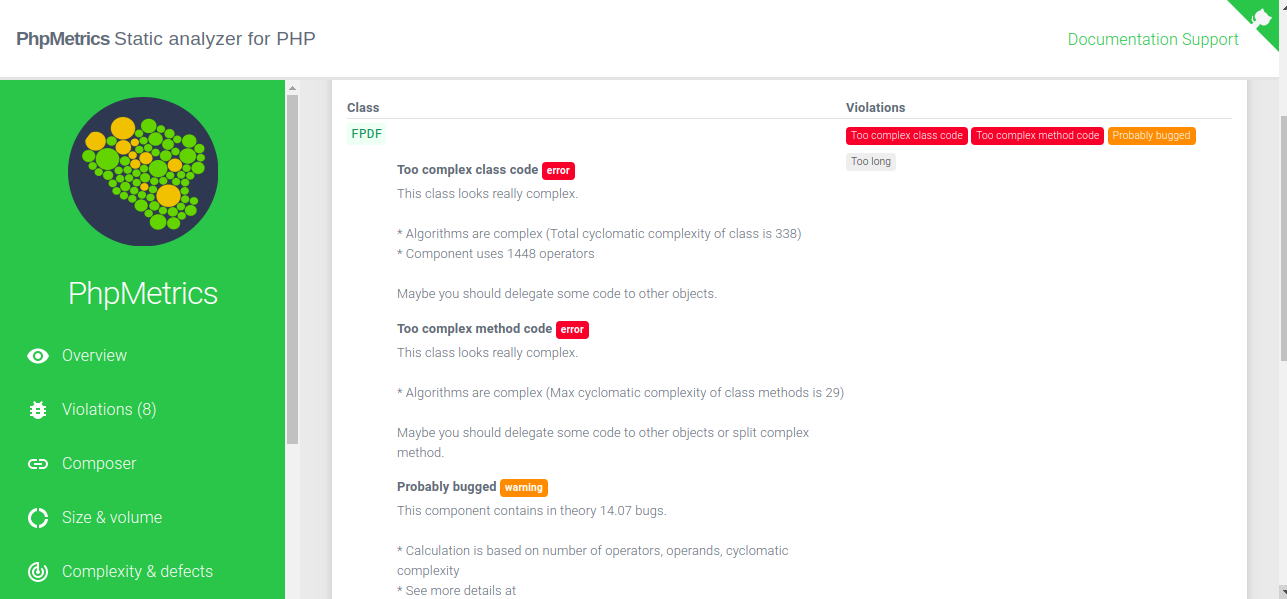
L’outil d’analyse PHP METRICS permet d’avoir des statistiques sur les mesures du code source. Il nous renvoie entre autres :

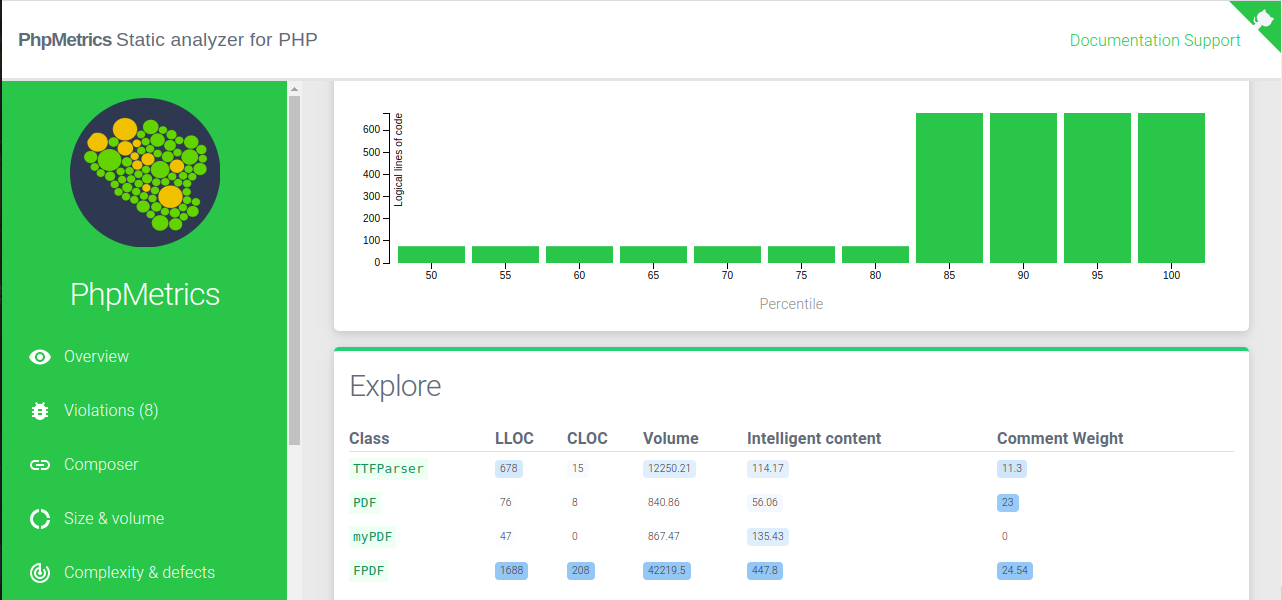
* Le nombre de violations ;
* Le nombre de lignes de code ;
* Le nombre de classe utilisée ;
* Le taux de complexité présente au sein de chaque classe ;
* L’utilisation des tests d’assertions ;
* Le taux de bug présent au sein de chaque classe.

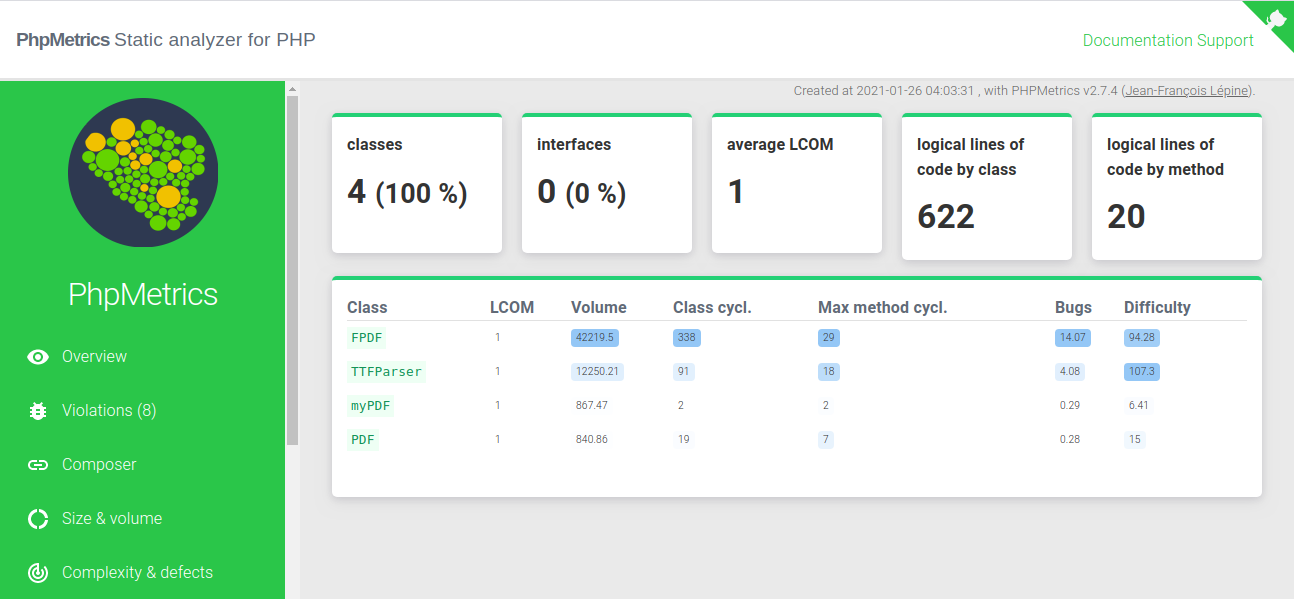
Il est à noter qu’à chaque résultat on y ajoutera des informations plus détaillé.

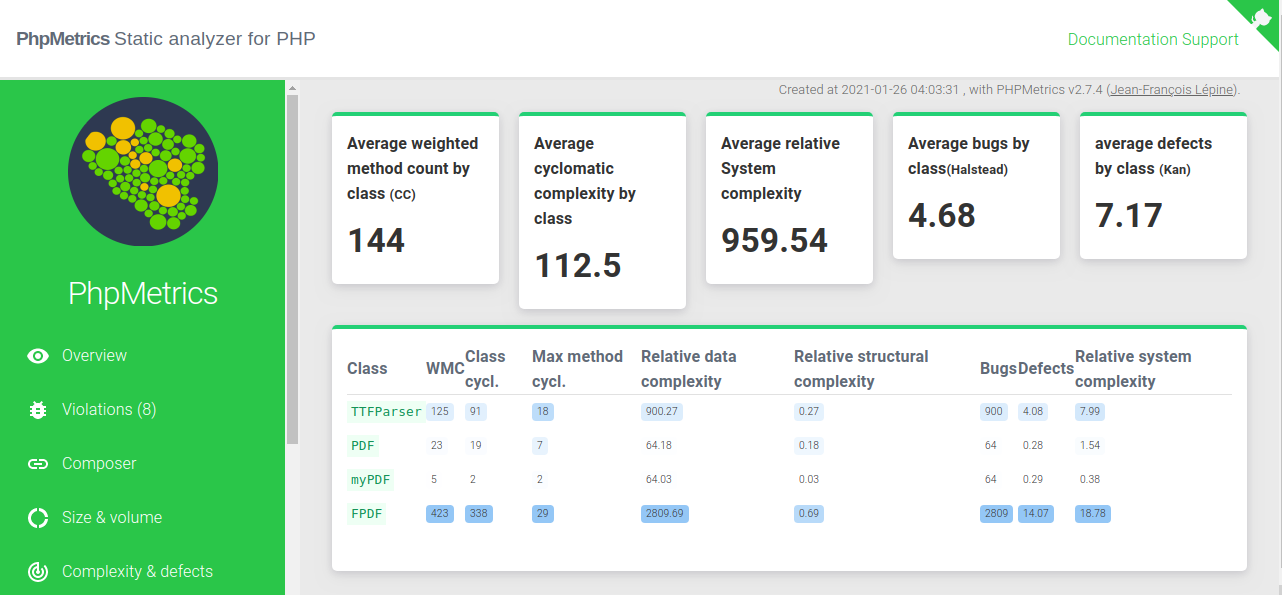
**Figure 1 :** Analyse globale de l’application avec PHP METRICS

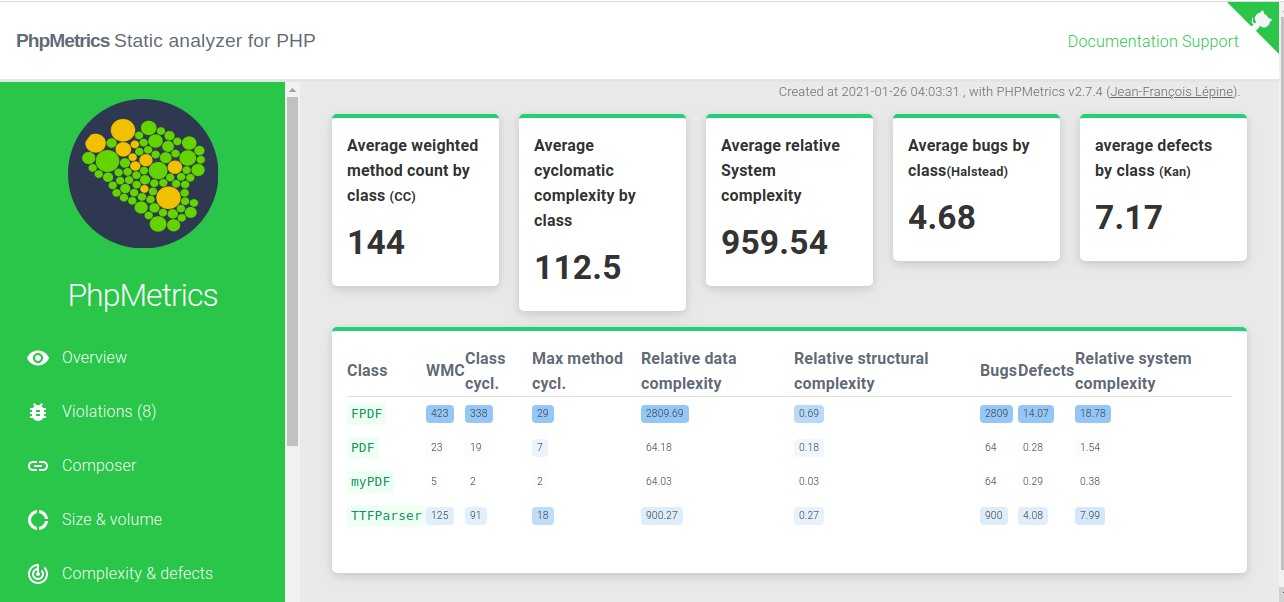
**Figure 2 :** Détails sur le nombre de violation

**Figure 3 :** Présentation des violations présentent dans la classe FPDF

**Figure 4 :** Détail sur les lignes de code ; Percentile distribution of logical lines of code by class

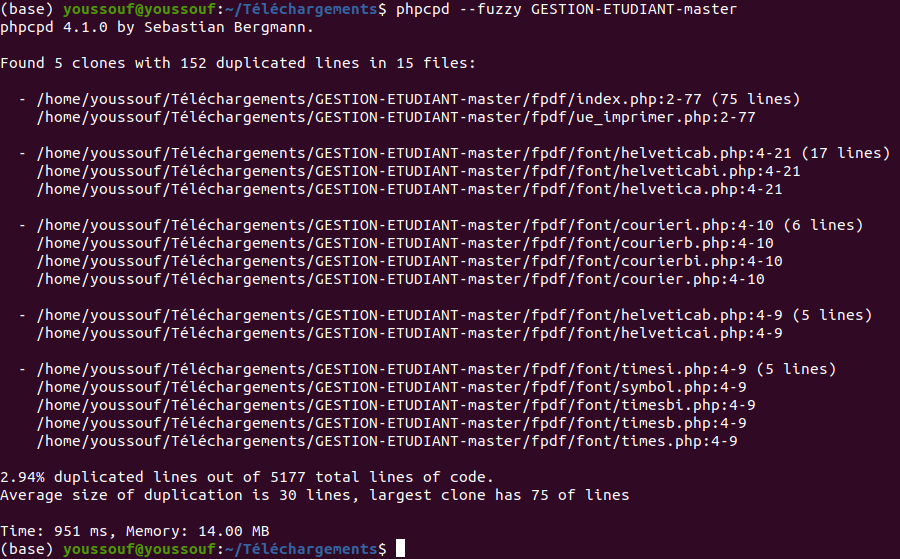
**Figure 5 :** Détail sur les classes

**Figure 6 :** Détail sur le [Average cyclomatic complexity by class](http://complexity.html/)

**Figure 7 :** Détail sur le taux de bugs présents dans chaque classe

Notons que cette outil renvoie par analyse le volume de ligne de code présente dans chaque classe, le poids des commentaires ; en bref un certain nombre d’informations.

# Résultat de l’outil d’analyse PHPCpd

 Le principal avantage de cet outil est qu’il nous renvoie les informations sur le nombre de ligne de code qui a été dupliqué, mais également sur le nombre de ligne de code présent dans le projet. Pour ce projet, voici un aperçu du résultat qui a été renvoyé.

**Figure 8 :** Résultat de l’analyse avec PHPCpd

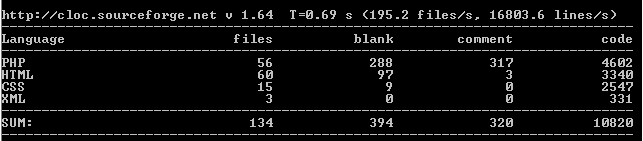
Nous constatons qu’avec cet outil, notre projet contient 5177 lignes de code, avec un taux de 2.94 % de lignes de code qui a été dupliquées. Cet outil nous renvoie les fichiers au sein desquels ces lignes de code ont été dupliquées, la taille moyenne de la duplication qui est de 30 lignes, mais également la taille du plus grand clone qui est de 75 lignes.

# Analyse de code avec l’outil CLOC

CLOC est un simple outil en ligne de commande qui compte les fichiers d’un projet par langage et pour chaque fichier, il compte les nombres de ligne de code, de ligne de commentaires, et de lignes vides.

Le résultat d’analyse de notre projet (Figure 9) nous montre que :

* Les langages utilisés sont PHP, HTML, CSS et XML ;
* Le projet compte 134 fichiers au total ;
* Le nombre de ligne de code source (SLOC) est de **10 000**



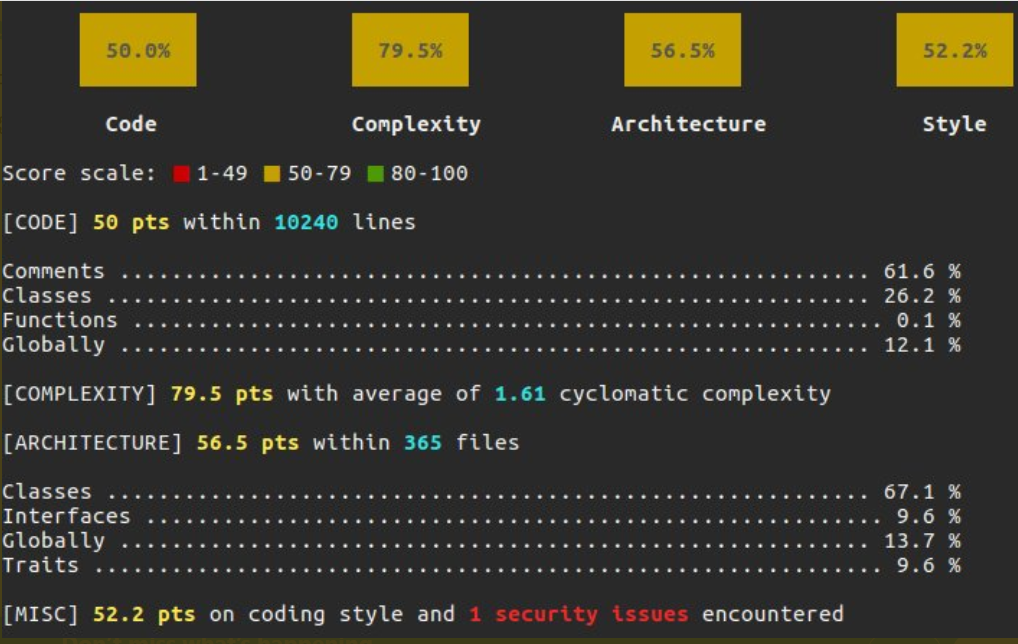
***Figure 9****: Résultat d’analyse du projet avec CLOC*

# Résultat de l’outil d’analyse PHPINSIGHTS

[**PHPInsights**](https://github.com/phpstan/phpstan) est un outil d’analyse de la qualité de code de vos projets PHP.

* PHP Insights analyse la qualité et le style de votre code. Il donne une belle vue d'ensemble de l'architecture du code et de sa complexité.
* PHP Insights s'adapte à votre framework : il est conçu pour fonctionner avec Laravel, Symfony, Yii, Wordpress, Magento2, etc.
* PHP Insights permet d'effectuer des tests de qualité : il rend votre code PHP fiable, simple et propre.

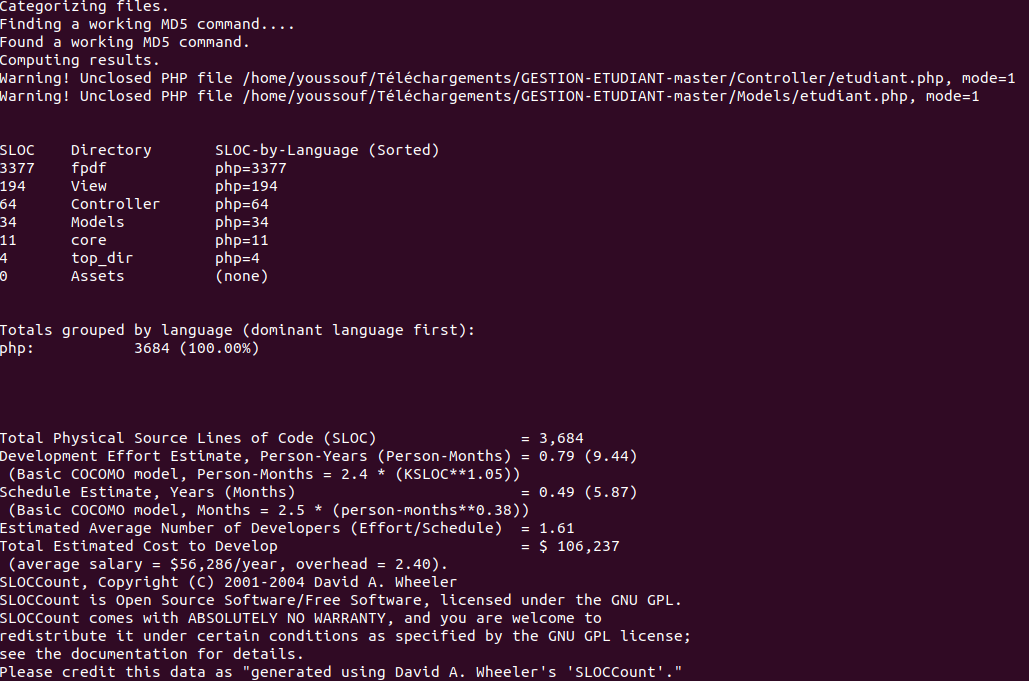
L’analyse de notre projet avec cet outil a produit le résultat présenté sur la figure ci-dessous :



**Figure 10** : Résultat de l’analyse du code avec [PHPInsights](https://github.com/phpstan/phpstan)

# Résultats de l’analyse de l’outil SLOCCOUNT

L’outil d’analyse sloccount a pour fonction principale de pouvoir effectuer une estimation du projet en utilisant la méthode COCOMO. En effet, cet outil renovoie la valeur de **l’Effort, le nombre de personne-mois**, mais également le nombre de ligne de code présente dans le code source. Le résultat de l’analyse du code en utilisant cet outil est le suivant :

**figure 11 :** Résultats de l’analyse avec l’outil d’analyse scloccount

Nous pouvons remarquer qu’en fonction des outils utilisés, le résultats d’analyse sont différents ; surtout au niveau de l’obtention du nombre de ligne de code source, vu que le compte s’effectue de déférente manière.

1. **Analyse de code avec l’outil Sonar**
   1. **Présentation de l’outil**

[**Sonar**](http://sonar.codehaus.org/) est un outil open source initialement développé par la société suisse [**Hortis**](http://www.hortis.ch/). Depuis novembre 2008, c'est la société suisse [**SonarSource**](http://www.sonarsource.com/) qui se charge du développement et du support de Sonar. Le but principal de cet outil est de fournir une analyse complète de la qualité d'une application en fournissant de nombreuses statistiques (ou **métriques**) sur ses projets. Ces données permettent ainsi d'évaluer la qualité du code, et d'en connaître l'évolution au cours du développement. Nous listons ici les principales fonctionnalités de l'outil Sonar.

* Tableau de bord complet des différents projets suivis.
* Détection rapide du code à risque.
* Mesures quantitatives : nombre de classes, duplication de code, etc.
* Mesures qualitatives : couverture et taux de réussite des tests, complexité du code, respect des règles de codage...
* Historiques des statistiques, pour en voir l'évolution au cours du temps.
* Support de plus de 600 règles de qualité.
* Gestion de profils pour les règles de codage.
* Visualisation du code source, surlignant les violations des règles de codage qui s'y trouvent.
* Fonction "Time machine" permettant de comparer plusieurs versions d'une même application.
* Identification des points faibles d'un projet.
* Support des plugins.

Par ailleurs, Sonar est multi-langage ; et disponible sous plusieurs formes :

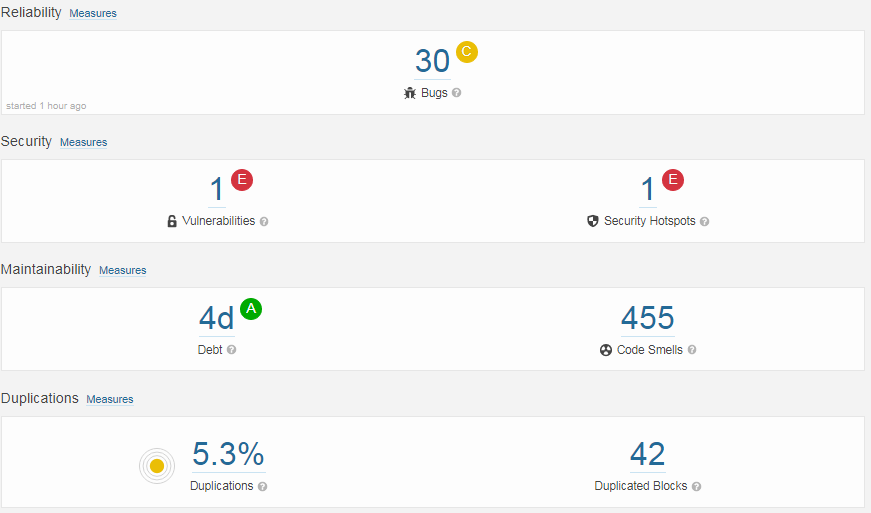
* Sous forme de plugin (**sonarLint**)
* Sous forme d’application indépendante (**SonarQube**)
* Sous forme d’application web (**sonarCloud**)

En outre, il peut être utilisé comme un outil d’intégration continu puisqu’il peut être relié à un dépôt comme **github**, **bitbuket** et bien d’autres. Et à chaque commit, il analyse le code afin de nous permettre de conserver la qualité de l’application durant son développement.

* 1. **Résultat de l’analyse du projet avec SonarCloud**

Nous avons opté pour notre analyse d’utiliser la version online de l’outil sonar (**sonarQube**) pour ne pas nous encombrer d’une installation locale. Nous l’avons directement relié avec notre dépôt github où le projet à analyser a été logé. Ainsi, une nouvelle analyse est lancée automatique à chaque commit.

L’analyse a produit un résultat dont la synthèse est représentée sur la figure suivante :

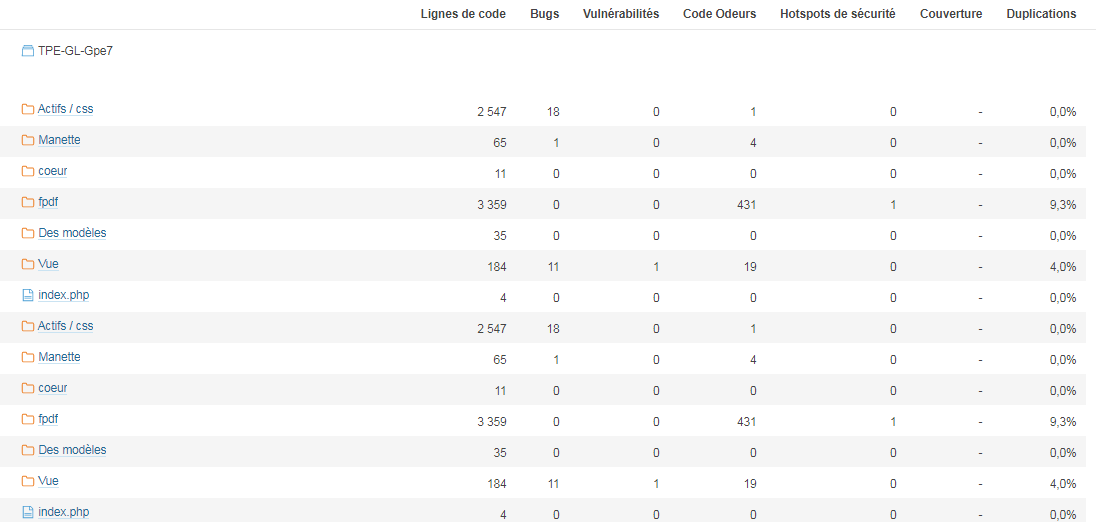


***Figure 12****: Résultat d’analyse du projet avec SonarCloud*

Il en ressort que

* Le projet contient **30 bugs**. La cote de fiabilité est **C** c’est-à-dire « *qu’il y a au moins un bug majeur ».*
* En termes de sécurité, sonar juge que l’application contient une vulnérabilité de blocage (la mention **E**).
* La mesure de la maintenabilité montre 4 jours de dette technique nécessaires pour réparer les **455 codes odeurs** (codes déroutant et difficiles à maintenir). La mention **A** signifie que « *le taux d’endettement technique est inférieur à 5% »*
* Enfin, **5,3%** de codessont **identique**, soit **42 bocks dupliqués**.

La capture suivante montre les statistiques (**lignes de code, bugs, vulnérabilités, codes odeurs, taux de duplication**) par dossier du projet.



***Figure 13****: Résultat d’analyse du projet par dossier avec SonarCloud*