








# Projet P\_App-CICD

Veronika Skupovska













FID2

## Un screenshot de l'exécution du pipeline principal

**build**  
succeeded 1 hour ago in 4m 51s

- >  Set up job
- >  Initialize containers
- >  Checkout code
- >  Setup php
- >  Cache Composer dependencies
- >  php dependencies
- >  Prepare Laravel
- >  Setup Node
- >  Install pnpm
- >  js dependencies
- >  Build frontend
- >  attendre MariaDB
- >  db config
- >  Run migrations
- >  php tests
- >  Upload test report

**security**  
succeeded 1 hour ago in 1m 24s

- >  Set up job
- >  Checkout code
- >  Initialize CodeQL
- >  Perform CodeQL Analysis
- >  Setup PHP
- >  Install PHPCS
- >  Run PHPCS
- >  PHPCS report
- >  Post Perform CodeQL Analysis
- >  Post Initialize CodeQL
- >  Post Checkout code
- >  Complete job

## Description

l'objectif était de mettre en place un pipeline CI/CD intégrant des outils de vérification statique de sécurité afin d'analyser automatiquement le code source lors de chaque modification.

## Outils disponibles sur le marché

Outil	Type	Avantages	Inconvénients
<b>Snyk</b>	Analyse de dépendances	Très précis, intégré aux CI	Fonctionnalités clés payantes
<b>SonarQube</b>	Analyse statique	Très complet, règles nombreuses	Version gratuite très limitée
<b>GitHub CodeQL</b>	Analyse statique	Gratuit, intégré à GitHub	Support PHP limité et contraignant
<b>PHP_CodeSniffer (PHPCS)</b>	Qualité / sécurité	Gratuit, simple, local	Analyse syntaxique, pas sémantique

## Choix effectués et justification

Les outils sélectionnés doivent être gratuits, simples à intégrer dans GitHub Actions, et compatibles avec les technologies utilisées ( PHP, JavaScript)

Au vu de ces contraintes, deux outils ont été choisis :

1. Code QL - analyse de sécurité du code JavaScript (dans le cadre du projet). permet de détecter automatiquement des vulnérabilités connues et des failles potentielles dans le code. Les résultats de cette analyse sont accessibles dans l'onglet Security / Code Scanning. Mais le support du PHP est incomplet et le dépôt doit être public ou disposer des permissions adaptées pour que l'analyse fonctionne correctement.
2. PHP\_CodeSniffer (PHPCS) - analyse statique du code PHP. Détecte les mauvaises pratiques de développement, les violations de standards de codage (PSR-12) et certains comportements à risque. Il est simple à utiliser, gratuit et

facilement intégrable dans un workflow GitHub Actions. Les résultats sont générés sous forme de rapport et joints au pipeline comme artefact

Les solutions Snyk et SonarQube ont été écartées. Bien que puissantes et reconnues, leurs fonctionnalités utiles pour une analyse de sécurité complète sont majoritairement payantes

## Analyse des résultats et gestion des faux positifs

Résultats PHP\_CodeSniffer :

- la gestion des espaces et des retours à la ligne (espaces manquants ou superflus)
- l'indentation incorrecte dans les structures multilignes
- le placement des parenthèses et des accolades
- des lignes dépassant la longueur maximale recommandée (120 caractères)

```
found" source="PSR12.Operators.OperatorSpacing.NoSpaceBefore"/>
<error line="61" column="96" severity="error" message="Expected at least 1 space after "."; 0
found" source="PSR12.Operators.OperatorSpacing.NoSpaceAfter"/>
<error line="66" column="1" severity="error" message="Blank line found at end of control
structure" source="Squiz.WhiteSpace.ControlStructureSpacing.SpacingBeforeClose"/>
<error line="75" column="92" severity="error" message="Blank line found at start of control
structure" source="Squiz.WhiteSpace.ControlStructureSpacing.SpacingAfterOpen"/>
<error line="84" column="95" severity="error" message="Expected at least 1 space before "."; 0
found" source="PSR12.Operators.OperatorSpacing.NoSpaceBefore"/>
<error line="84" column="95" severity="error" message="Expected at least 1 space after "."; 0
found" source="PSR12.Operators.OperatorSpacing.NoSpaceAfter"/>
<error line="89" column="1" severity="error" message="Blank line found at end of control
structure" source="Squiz.WhiteSpace.ControlStructureSpacing.SpacingBeforeClose"/>
<error line="138" column="1" severity="warning" message="Line exceeds 120 characters; contains
180 characters" source="Generic.Files.LineLength.TooLong"/>
<error line="156" column="105" severity="warning" message="Line exceeds 120 characters; contains
```

donc les alertes sur non-uniformité du style de code

Résultats CodeQL :

- utilisation de générateurs de nombres aléatoires non sécurisés
- problèmes d'échappement ou d'encodage incomplet de chaînes de caractères

<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Incomplete string escaping or encoding</b>	High	Library	#5 closed as fixed 1 hour ago • Detected by CodeQL in vendor/.../dist/index.esm.js :1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Incomplete string escaping or encoding</b>	High		#4 closed as fixed 1 hour ago • Detected by CodeQL in public/.../assets/apps-BQkWFN5d.js :31
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Incomplete string escaping or encoding</b>	High		#3 closed as fixed 1 hour ago • Detected by CodeQL in public/.../assets/apps-BQkWFN5d.js :31
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Insecure randomness</b>	High		#2 closed as fixed 1 hour ago • Detected by CodeQL in public/.../assets/dropzone-Cafc4ooc.js :22
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Prototype-polluting function</b>	Medium		#13 closed as fixed 1 hour ago • Detected by CodeQL in public/.../assets/echarts-core-1eLjUTDF.js :14
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<b>Overly permissive regular expression range</b>	Medium	Library	#1 closed as fixed 1 hour ago • Detected by CodeQL in vendor/.../highlightjs/highlight.pack.js :397

Une part significative de ces alertes concerne des fichiers situés dans les répertoires vendor et public/assets

## Structure du pipeline

Le pipeline CI/CD mis en place est composé de deux jobs :

- build est chargé de vérifier que l'application peut être installée, configurée et exécutée correctement.  
Il effectue le checkout du code source, configure les environnements PHP et Node.js, installe les dépendances PHP et JavaScript, génère le build du frontend. Initialise MariaDB, les migrations sont exécutées et les tests PHP sont lancés à l'aide de l'outil Pest. Les résultats des tests sont sauvegardés sous forme d'artefact afin de pouvoir être consultés après l'exécution du pipeline.
- security est exécuté uniquement si le job *build* s'est terminé avec succès  
Il se concentre sur l'analyse statique du code. CodeQL est utilisé pour analyser le code JavaScript, tandis que PHP\_CodeSniffer analyse le code PHP. Les rapports générés sont également stockés sous forme d'artefacts.

## avantages / inconvénients

Scan code and beyond for any exposed secrets

## Conclusion

Ce projet a permis de comprendre le fonctionnement d'un pipeline CI/CD, et de se familiariser avec les outils de sécurité automatisés et de confronter les limites des solutions gratuites disponibles sur le marché. Malgré de nombreuses difficultés liées aux permissions GitHub, au support des langages, versions, à la configuration des tests et à la gestion des artefacts, le workflow est fonctionnel.

