Pipeline CI/CD & upgrade dépendances PAPIE

Migration Symfony 6.2 → 6.4, sécurisation applicative et mise en place du pipeline CI/CD (préprod)

Développeur	COUR Cyrille
Organisation	UPV — Union Patronale du Var
Formation	BTS SIO SLAM — Alternance 2025/2027
Stack	Symfony 6.4 / PHP 8.1
Date	14/08/2026
Statut	CI/CD préprod validé et fonctionnel sur main — déploiement prod encore manuel

Sommaire

Sommaire 2
1. Contexte 3
2. Environnement 3
2.1 Prérequis 3
2.2 Environnements 3
3. Procédure 3
3.1 Migration Symfony 6.2 → 6.4 3
3.2 Correctifs de sécurité appliqués 4
3.2.1 Suppression du flux « mot de passe oublié » 4
3.2.2 Open redirect / XSS réfléchi via le paramètre referer 4
3.2.3 Absence de vérification CSRF sur la suppression différée 4
3.2.4 Fuite mémoire sur les listes de personnes 4
3.3 Suite de tests automatisés 4
3.4 Pipeline CI/CD — Préproduction 5
3.4.1 Job « test » — audit et vérifications 5
3.4.2 Job « deploy » — construction et déploiement 5
3.5 Secrets et configuration 5
4. Structure / Données 6
4.1 Résumé de l'état actuel 6
4.2 Détail de la suite de tests 7
5. Déploiement 7
5.1 Préproduction — automatisée 7
5.2 Production — état actuel (manuel) 7
6. Points de vigilance 8
6.1 Prochaines étapes 8

1. Contexte

Papie est une application Symfony 6 / PHP 8.1 utilisée par des travailleurs sociaux pour gérer des dossiers de personnes physiques et morales, des consultations et des interventions. L'application est en production active, avec un environnement de préproduction dédié (preprod-papie.upv.org) et une migration de données en cours depuis un ancien système (Metycea).

Objectifs initiaux de ce chantier :

- S'assurer que les dépendances applicatives ne contiennent pas de vulnérabilité de sécurité connue (composer audit).
- Auditer le code applicatif avec un outil d'analyse statique (Semgrep).
- Mettre à niveau Symfony (6.2 → 6.4) et les dépendances associées (Twig 1/2 → 3).
- Fiabiliser le pipeline de déploiement vers la préproduction (GitHub Actions), jusque-là manuel et sans aucune vérification avant mise en ligne.

Note — Ce chantier couvre la mise à niveau du framework, la correction de plusieurs failles de sécurité, la construction d'une suite de tests automatisés et la mise en place d'un pipeline CI/CD pour la préproduction. Le déploiement en production reste à ce stade manuel (voir §5).

2. Environnement

2.1 Prérequis

- Symfony 6.4 (LTS) / PHP 8.1
- Twig 3.x
- Composer (audit des dépendances)
- Semgrep (analyse statique)
- GitHub Actions (CI/CD)
- Base MySQL éphémère dédiée aux tests CI

2.2 Environnements

- Préproduction : preprod-papie.upv.org — APP_ENV=dev (WebProfilerBundle actif)
- Production : APP_ENV=prod — déploiement manuel via WinSCP à ce jour

3. Procédure

3.1 Migration Symfony 6.2 → 6.4

Mise à niveau du framework Symfony de la version 6.2 vers la version 6.4 (LTS), avec migration de Twig en version 3.x au passage. Aucune régression détectée à l'issue de la migration.

3.2 Correctifs de sécurité appliqués

3.2.1 Suppression du flux « mot de passe oublié »

Constat : le contrôleur générait un nouveau mot de passe et l'enregistrait en base sans jamais l'envoyer par e-mail (le mailer configurer n'était jamais sollicité). Un tiers connaissant l'adresse e-mail d'un compte pouvait ainsi déclencher un changement de mot de passe et bloquer l'accès du titulaire, sans authentification préalable.

Décision : la gestion des mots de passe étant assurée manuellement par l'équipe, la fonctionnalité a été jugée inutile et désactivée entièrement (route, contrôleur, lien depuis l'écran de connexion).

3.2.2 Open redirect / XSS réfléchi via le paramètre referer

Constat (détecté par Semgrep) : cinq écrans d'administration lisaient un paramètre referer transmis par l'URL et l'utilisaient tel quel pour rediriger l'utilisateur après une action, sans validation. Une URL forgée pouvait rediriger un utilisateur authentifié vers un site externe, ou injecter un lien de type javascript:.

Correctif : validation centralisée dans appendReferer(), qui n'accepte désormais qu'un chemin interne résolvable par le routeur Symfony (router->match()). Toute valeur non conforme retombe sur la route d'index. Les points de lecture directe qui contournaient le trait ont été alignés sur la même validation.

3.2.3 Absence de vérification CSRF sur la suppression différée

Constat : la fonction gérant la suppression différée des personnes physiques, morales et des dossiers (ainsi que la réouverture de dossier) acceptait une requête POST sans jamais vérifier de jeton CSRF, contrairement au circuit de suppression standard qui, lui, était protégé.

Risque : un tiers pouvait forger une page piégée déclenchant cette action à l'insu d'un administrateur connecté. Le paramètre cookie_samesite: lax déjà en place limitait fortement l'exploitation réelle, mais aucune protection applicative ne venait renforcer cette seule barrière.

Correctif : vérification du jeton CSRF ajoutée, alignée sur le circuit déjà protégé, avec un message d'erreur explicite affiché à l'utilisateur en cas de jeton invalide (plutôt qu'un échec silencieux).

3.2.4 Fuite mémoire sur les listes de personnes

Constat : les écrans « Personnes physiques » et « Personnes morales » provoquaient un dépassement de la limite mémoire PHP (128 Mo) en préproduction. Cause : chaque ligne de la liste déclenchait le rendu du formulaire de suppression via une sous-requête Symfony complète (render(controller(...))), profilée intégralement par la barre de débogage. Sur une page de 100 lignes, cela représentait 101 sous-requêtes profilées.

Correctif : les formulaires de suppression sont désormais rendus directement dans la vue via une macro Twig, sans passer par une sous-requête. Dix-sept templates concernés. Vérifié : rendu HTML identique, jeton CSRF valide, requête de suppression fonctionnelle de bout en bout.

3.3 Suite de tests automatisés

La suite est passée de 2 tests (couverture quasi nulle) à 123 tests et 300 assertions. Elle est exécutée automatiquement à chaque push sur main via le pipeline CI/CD et bloque le déploiement en cas d'échec.

3.4 Pipeline CI/CD — Préproduction

Le pipeline est déclenché automatiquement à chaque push sur la branche main. Il se décompose en deux jobs GitHub Actions : le premier vérifie la qualité et la sécurité du code, le second ne s'exécute que si le premier réussit et déploie sur le serveur de préproduction. Ce pipeline constitue désormais la norme à décliner sur les autres applications du portfolio.

3.4.1 Job « test » — audit et vérifications

Exécuté sur une machine GitHub Actions temporaire (jetable), avec une base MySQL éphémère dédiée — aucune interaction avec les bases de données locales ou de préproduction.

```
composer audit          # vulnérabilités connues — bloquant
php bin/phpunit         # suite de tests complète — bloquant
semgrep --config=auto   # analyse statique — informatif, non bloquant
```

Si l'une des étapes bloquantes échoue, le déploiement n'a pas lieu.

3.4.2 Job « deploy » — construction et déploiement

S'exécute uniquement si le job « test » a réussi (needs: test).

- Installation des dépendances PHP — composer install (avec dépendances de dev, la préproduction tournant volontairement en APP_ENV=dev et nécessitant WebProfilerBundle), construit sur la machine GitHub, jamais sur le serveur.
- Construction des assets — npm ci puis npm run build (Webpack Encore).
- Déploiement du code source — connexion SSH au serveur, git fetch puis git reset --hard origin/main, suivi d'un vidage du cache (rm -rf var/cache/*).
- Synchronisation de vendor/ — transféré par SCP depuis la machine GitHub (le serveur de préproduction ne peut pas exécuter composer lui-même, voir §6).
- Synchronisation de public/build/ — assets compilés, transférés par SCP.
- Vérification finale — requête HTTP sur la préproduction ; le pipeline échoue si le code retour est ≥ 500.

3.5 Secrets et configuration

- SSH_PRIVATE_KEY — secret GitHub, clé privée SSH pour la connexion au serveur de préproduction (port 2222).
- .env (job test) — fichier temporaire créé à la volée avec APP_ENV=test ; jamais transféré au serveur.
- .env (job deploy) — fichier temporaire créé à la volée avec APP_ENV=prod, uniquement pour permettre l'exécution de composer install ; jamais transféré au serveur.
- .env réel (serveur) — jamais touché, reste en permanence sur le serveur, gitignoré, contient la configuration réelle (APP_ENV=dev, titre, base de données).

4. Structure / Données

4.1 Résumé de l'état actuel

Élément	Statut	Remarque
Migration Symfony 6.2 → 6.4	Terminé	Twig migré en 3.x au passage. Aucune régression détectée.
Suite de tests automatisés	123 tests / 300 assertions	Partie de 2 tests initiaux ; couvre CRUD, suppression, workflow, partage, CSRF.
composer audit	Aucune vulnérabilité	2 packages abandonnés signalés (doctrine/annotations, doctrine/cache) — non bloquant.
Semgrep (analyse statique)	Traité	118 findings triés, 4 % de signal réel ; correctif appliqué (open redirect referer).
CI/CD préprod (GitHub Actions)	Fonctionnel sur main	Audit + tests + build + déploiement automatique à chaque push.
CI/CD production	Non fait	Déploiement toujours manuel via WinSCP. À automatiser (voir §5).
Faille mot de passe oublié	Corrigée	Flux désactivé et retiré (fonctionnalité jugée inutile en interne).
Faille open redirect / XSS (referer)	Corrigée	Validation centralisée via router->match() dans appendReferer().
Faille CSRF (suppression différée)	Corrigée	Vérification de token ajoutée dans deleteDiffFunc() et reopenAction().
Fuite mémoire pages Personnes	Corrigée	101 sous-requêtes profilées ramenées à 1 via une macro Twig.
Migration Symfony 7	Non planifiée	6.4 est LTS. Un point bloquant déjà identifié (InterventionType). Pas urgent.

4.2 Détail de la suite de tests

Classe de test	Catégorie	Couverture
AdminSmokeTest	Fumée / non-régression	Vérifie que chaque route /admin répond sans erreur serveur et exige une authentification.
DeletionTest	Suppression	Suppression différée des personnes physiques, morales, dossiers ; bascule et liste de suppression.
CrudTest	Création / modification	Création et modification de personnes physiques et morales, avec cas de rejet (SIRET, téléphone).
ConsultationCreationTest	Création de dossier	Parcours en deux temps (brouillon puis dossier complet), avec contre-épreuve sur motif manquant.
ConsultationWorkflowTest	Cycle de vie du dossier	Statut, clôture, réouverture, réaffectation, lecture seule, gel sur statut « à réaffecter ».
InterventionCrudTest	Interventions	Saisie, cohérence personne/dossier, modification, bornes de date, refus sur dossier clôturé.
SharingTest	Partage de dossier	Partage, accès gagné/refusé, révocation, restrictions du propriétaire pendant un partage actif.
CsrfProtectionTest	Protection CSRF	Vérifie le rejet des requêtes de suppression et de réouverture sans jeton valide.

5. Déploiement

5.1 Préproduction — automatisée

Le déploiement en préproduction est entièrement automatisé via le pipeline CI/CD décrit en §3.4 : audit, tests, build et déploiement s'enchaînent à chaque push sur main, sans intervention manuelle.

5.2 Production — état actuel (manuel)

Le déploiement en production reste, à ce jour, manuel. Aucune automatisation n'a encore été mise en place, par prudence : la production est utilisée en conditions réelles et une automatisation mal maîtrisée y présenterait un risque plus élevé qu'en préproduction.

Différences connues à respecter par rapport à la préproduction, essentielles si une automatisation est mise en place :

- APP_ENV=prod en production (contre dev en préproduction) — le dossier vendor/ doit donc être construit avec l'option --no-dev, sans quoi WebProfilerBundle provoque une erreur fatale.
- Titre de l'application distinct entre les deux environnements (variable APP_TITLE, définie séparément dans le .env de chaque serveur).
- Version PHP à reconfirmer côté production avant toute automatisation (8.1.34 confirmée côté préproduction via phpContainer, à vérifier côté production).

Note — Piste retenue pour l'automatisation future (non mise en œuvre à ce jour) : un second workflow GitHub Actions, déclenché manuellement (workflow_dispatch) plutôt qu'automatiquement sur push, reprenant la même mécanique que la préproduction avec composer install --no-dev et les identifiants propres au serveur de production. Cette automatisation sera documentée ici une fois mise en place et validée.

6. Points de vigilance

Attention — Le PHP en ligne de commande du serveur de préproduction n'est accessible qu'au travers d'un script d'encapsulation (phpContainer, exécuté via sudo), dont la configuration désactive la fonction putenv(). Or Composer et le composant Symfony Runtime en dépendent pour s'exécuter correctement : il est donc impossible de lancer composer install directement sur ce serveur en l'état. Conséquence retenue : le dossier vendor/ est systématiquement construit sur la machine GitHub Actions puis transféré déjà prêt vers le serveur. Piste d'amélioration non traitée : lever la restriction putenv() sur phpContainer nécessiterait un accès root (à voir avec l'hébergeur).

Attention — Semgrep est exécuté en mode informatif (non bloquant) sur ce pipeline, compte tenu d'un fort taux de faux positifs sur les règles génériques utilisées (118 findings triés pour 4 % de signal réel). Ne pas le rendre bloquant sans un jeu de règles affiné.

6.1 Prochaines étapes

- Automatiser le déploiement en production (workflow manuel dédié, voir §5.2).
- Étendre ce modèle de pipeline CI/CD aux autres applications du portfolio, une fois qu'un premier déploiement production automatisé aura validé l'approche.
- Réévaluer la migration vers Symfony 7 (point bloquant déjà identifié : InterventionType).