VEILLE TECHNOLOGIQUE - Symfony

Etat des lieux :

Dans une entreprise comme ARES Formation, ayant créé un ENT propre à l'école, "PPAP", pour des formations post-bac, le framework utilisé est Symfony 7.3 avec PHP 8.4 pour l'application web. L'application gère des plannings, de nombreux formulaires (suivis de cours et personnels pour chaque utilisateur, import de factures selon une période donnée), créations de devoir avec gestion des notes, création de pdf pour des bulletins, des contrats entre l'école et les professeurs.
De plus, l'application a un visuel pour les tuteurs d'entreprise des élèves en alternance avec des statistiques, informations récupérées pour son alternant.
PPAP se base par ailleurs énormément sur des dépendances et composants installés via la commande : "composer require".

Présentation générale de Symfony :

Symfony est un framework PHP basé sur l'architecture MVC (Modèle - Vue - Controller). Il a été créé par SensioLabs et est très apprécié par les développeurs pour le développement d'applications web en particulier. La version 5.4 est la version la plus ancienne encore stable (fin des correctifs de bugs en 2024 avec une fin des correctifs de sécurité en 2029). Depuis Symfony 6, PHP 7 n'est plus supportées, uniquement PHP ≥ 8.

Officiellement Symfony 7.4.0 et Symfony 8.0.0 sont sortis le 27 novembre 2025, Symfony 7.4.0 étant une version LTS (support à long terme, assurée de tenir sur plusieurs années).

Pour rappel, un framework est un ensemble d'outils, composants permettant de :

- structurer une application
- Développer efficacement
- Eviter le code redondant
- Utiliser des normes de programmation

Caractéristique	Symfony 7.4 (LTS)	Symfony 8.0
Statut de support	Long-Term Support (LTS)	Standard
Prérequis PHP	≥ 8.2.0	≥ 8.4.0
Fin des corrections (Bugs)	Novembre 2028	Juillet 2026
Fin du support Sécurité	Novembre 2029	Juillet 2026
Dépréciations ?	Présentes	Supprimée

Dernières versions de Symfony :

Symfony 8.0.8 & Symfony 7.4.8 : 1 avril 2026

Changements de Symfony 7.4 à surveiller pour une application comme PPAP

1. Éléments dépréciés (à surveiller sur PPAP)

De nombreuses méthodes sont dépréciées en Symfony 7.4 afin d'être supprimé pour la version 8.0. Le but de cette version est donc de préparer à la transition vers 8.0. Certains éléments sont à surveiller dans une application comme PPAP pour ARES Formation :

Composant	Ce qui change	Migration
Routing	Format de configuration XML déprécié	Convertir en YAML ou PHP.
Routing	getEnv() et setEnv() sur Router supprimés.Définitivement retirées de la classe Router.	Utiliser les variables d'environnement Symfony standards (%env(VAR)%).
DoctrineBridge	AbstractDoctrineExtension déprécié, de même que l'auto-mapping d'entités Doctrine (Doctrine entity auto-mapping).	Déclarer explicitement les entités dans la config Doctrine.
HttpFoundation	La méthode Request::get() qui cherchait dans attributes, query et request en même temps est dépréciée.	Utiliser explicitement $request->attributes->get(), $request->query->get() ou $request->request->get().

Certaines dépreciations sont externes à Symfony mais concernent leur composants et sont importes à prendre en compte puisqu'elles migrent en même temps que Symfony, par exemple :

Composant	**Dépréciation**	**Avant**	**Après**	**Notes**
Doctrine Collections	Criteria::ASC / Criteria::DESC	->orderBy(['start' => Criteria::ASC])	->orderBy(['start' => Order::Ascending])	Externe à Symfony mais impacte tous les projets Symfony/Doctrine. Enum Order dispo depuis Doctrine Collections 2.x

De nombreux Criteria sont utilisés dans des projets Symfony et en particulier sur PPAP, ce n'est pas négigeable.

2. Nouveautés de Symfony 7.4

https://symfony.com/blog/symfony-7-4-curated-new-features

Certaines nouveautés de Symfony 7.4 peuvent être intéressantes à mettre en place dans un ENT (Espace numérique de travail) interne à l'école :

Version	Composant	Fonctionnalité
7.4	Form	FormFlow, formulaires multi-étapes natifs Gestion native des étapes, états et validations intermédiaires.
7.4	Security	Nouvel attribut PHP #[IsSignatureValid] pour valider automatiquement la signature d'une URI dans un contrôleur (tout lien à usage unique). Possibilité de filtrer par méthode (methods: ['POST', 'PUT'])]
7.4	Routing	Les attributs #[Route] sont désormais détectés dans n'importe quel répertoire de l'application (pas seulement src/Controller).
7.4	Video Constraint	Nouvelle contrainte pour valider des fichiers vidéo (dimensions, codecs, formats)

En détails, le formflow va permettre de créer automatiquement les formulaires multi-étapes (aussi appelés wizards) très facilement. Symfony va savoir gérer et implémenter chaque étape :
https://symfony.com/blog/new-in-symfony-7-4-multi-step-forms

```php
use Symfony\Component\Form\Flow\AbstractFlowType;

class UserSignUpType extends AbstractFlowType
{
    public function buildFormFlow(FormFlowBuilderInterface $builder, array $options): void
    {
        $builder->addStep('personal', UserSignUpPersonalType::class);
        $builder->addStep('professional', UserSignUpProfessionalType::class);
        $builder->addStep('account', UserSignUpAccountType::class);

        $builder->add('navigator', NavigatorFlowType::class);
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => UserSignUp::class,
            // `$currentStep` is a property defined in the UserSignUp class
            'step_property_path' => 'currentStep',
        ]);
    }
```

Quant à l'attribut IsSignatureValid, elle va permettre de s'assurer qu'un lien unique n'a pas été falsifié beaucoup plus simplement. La signature garantit que l'URL n'a pas été falsifiée.

https://symfony.com/blog/new-in-symfony-7-4-attribute-improvements

```php
use App\Security\Attribute\IsSignatureValid;

class SomeController extends AbstractController
{
    #[IsSignatureValid]
    public function someAction(): Response
    {
        // ...
    }

    // you can also check signatures for specific HTTP methods
    #[IsSignatureValid(methods: ['POST', 'PUT'])]
    public function updateItem(): Response
    {
        // ...
    }
}
```

Enfin, les contraintes vidéos vont ressembler à :

```php
namespace App\Entity;

use Symfony\Component\HttpFoundation\File\File;
use Symfony\Component\Validator\Constraints as Assert;

class VideoContent
{
    #[Assert\Video(
        maxWidth: 1920,
        maxHeight: 1080,
        maxSize: '100M',
        mimeTypes: ['video/mp4', 'video/webm'],
    )]
    private File $videoFile;
}
```

The constraint supports many validation options, including `minWidth`, `maxWidth`, `minHeight`, `maxHeight`, `minRatio`, `maxRatio`, `minPixels`, and `maxPixels`. You can also control video orientation by disabling landscape, portrait, or square videos:

```php
#[Assert\Video(
    minWidth: 1280,
    minHeight: 720,
    allowPortrait: false,
    allowSquare: false,
)]
private File $videoFile;
```

By default, the constraint accepts common video codecs and container formats such as H.264, HEVC, VP9, AV1, MP4, WebM, and MKV. You can customize these lists to fit your own requirements:

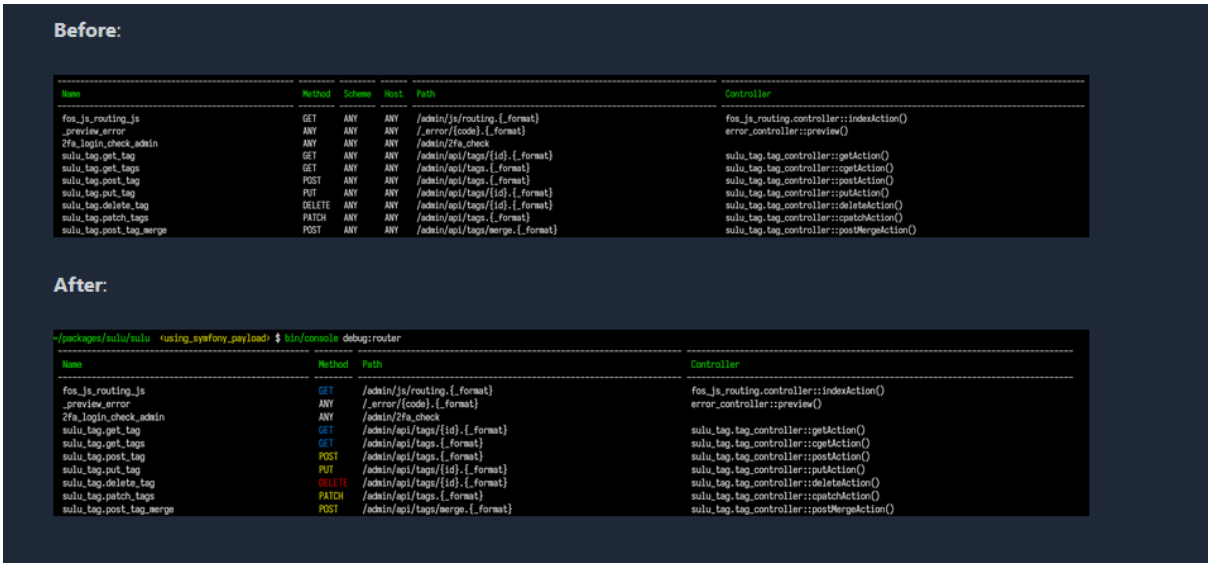
```php
#[Assert\Video(
    allowedCodecs: ['h264', 'hevc'],
    allowedContainers: ['mp4', 'webm'],
)]
private File $videoFile;
```

C'est l'équivalent de #[Assert\Image] qui existe depuis Symfony 2.1 mais pour les vidéos. Il permet de géré nativement les dimensions minimales, l'orientation de la vidéo, codecs autorisés (h264, hevc), les formats autorisés (mp4, webm)...

3. Expérience Développeur (DX)

https://symfony.com/blog/new-in-symfony-7-4-dx-improvements-part-2

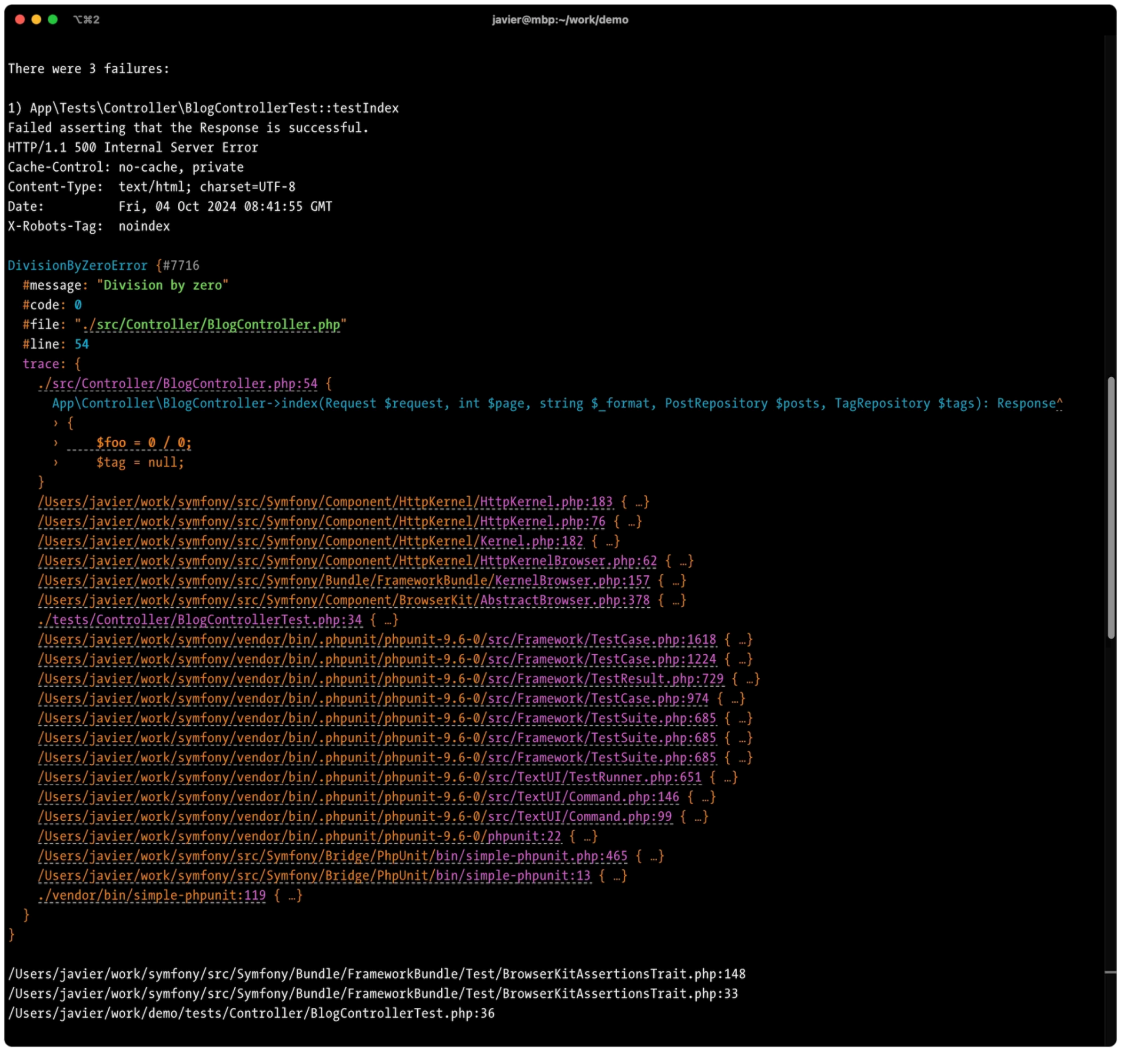
- debug:router amélioré (couleurs, scheme et host affichés que si différents de “any” + de clarté). Un debug plus visuel pour le développeur



- Les Dumps : (dump() et dd()) affichent les exceptions dans le terminal en texte brut (plutôt qu’en HTML). Les tests sont plus lisibles et plus clairs https://symfony.com/blog/new-in-symfony-7-4-better-exceptions-in-terminal

Avant :

Après :



- Intégration native du mode “worker” de FrankenPHP, plus besoin d’utiliser runtime/frankenphp-symfony pour le mode worker.
  kernel.default_locale utilisé par défaut hors des requêtes HTTP

On considère aujourd’hui environ 13000 lignes de codes supprimées pour Symfony 8.

Problématique en entreprise :

PPAP utilise PHP 8.4 avec Symfony 7.3. Cette version est actuellement une des dernières versions stables en date. Cependant, la question du changement de version se pose avec l’arrivée de Symfony 7.4 et Symfony 8.0 en novembre 2025.

La version 7.3 de Symfony, bien que stable, n'est pas une version LTS, ce qui signifie que son support prend fin en janvier 2026. Le manque de mise à jour et de support peut alors poser problème pour une entreprise comme ARES Formation pour plusieurs raisons :

- Une école a de nouveaux étudiants tous les ans, ce qui signifie de nouvelles données personnelles et confidentielles.
- Un site avec une version non maintenue voire obsolète sur le long terme peut voir de nombreuses failles de sécurité jamais corrigées et donc mettre à risque les données enregistrées.

Des questions se posent alors :

- Est-ce pertinent de faire une montée de version ?
- Est-ce pertinent de faire de petites mises à jour mineures régulièrement plutôt que de grosses montées de versions ?
- Est-ce pertinent de rester sur Symfony ?

Comparatif de solutions

Plusieurs solutions peuvent être proposées dès l'arrivée de la fin de maintenance de Symfony 7.3 :

1. Changer complètement de framework.

Entre tous les choix de framework, il serait intéressant de choisir un framework proche en architecture de Symfony comme Laravel.
Laravel est construit sur la même architecture que Symfony (MVC). Cependant, ils ont des structures différentes :

- Les deux frameworks n'ont pas le même générateur de vue (Symfony utilise Twig, Laravel utilise Blade).
- Symfony est très indépendant et maniable avec des "packages" comparé à Laravel qui possède beaucoup de composants externes (parfois proches de Symfony).
- Laravel est mis à jour moins régulièrement que Symfony : Laravel sort des versions environ une fois par an : La version 12 est sortie en février 2025 tandis que la version 11 est sortie en mars 2024. Symfony a des mises à jour

environ tous les 6-7 mois : la version 7.3 est sortie en mai 2025 tandis que la 7.4 sort en novembre 2025 (LTS) avec la 8.0.

Sur un projet contenant des données confidentielles et sensibles, il est peu recommandé de changer complètement de framework puisque c'est une perte de temps et un gros risque pour les données. Cela pourrait aussi avoir un coût, (comme engager un développeur compétent en Laravel)
Laravel étant un framework avec des fonctionnalités différentes, des packages externes, cela pourrait être une perte de temps immense de réécrire, d'adapter tout le code, de repenser à une nouvelle méthode de travail et pourrait perturber les utilisateurs.

2. Rester sur Symfony 7.3 voire régresser :

La deuxième solution serait pour l'application de rester en Symfony 7.3. Cependant, la fin de support de Symfony 7.3 est en janvier 2026, plus aucun bug, faille de sécurité ou autre ne sera corrigé, cela peut mettre à risque des données. De plus, plus on attend longtemps avant de mettre à jour, plus les écarts et les changements entre chaque version (majeure et mineure) vont s'agrandir et la montée de version sera compliquée et longue (une potentielle plus grosse perte de temps sur le long terme, que de suivre régulièrement les avancées).
Enfin, régresser de version (comme retourner en 7.2) signifierait que certaines parties du code devraient être réécrites si l'application utilise des nouveautés spécifiques à Symfony 7.3.

3. Une autre solution proposée serait donc de faire la mise à jour en Symfony 7.4 dans sa sortie qui est une version LTS.

Une version LTS signifie un support de plusieurs années (pour Symfony 7.4, la correction de bugs est prévue jusqu'en novembre 2028 avec la correction de problèmes de sécurité jusqu'en novembre 2029). Une version LTS permet une sécurité du code sur plusieurs années avec une certitude de corrections de bugs. De plus, faire la mise à jour en Symfony 7.4 permet à l'application de se tenir à jour et avoir la possibilité de faire des mises à jour mineures régulièrement sans prendre trop de risque concernant les données exploitées (de Symfony 7.4 à Symfony 7.4.1 par exemple). Enfin, la mise à jour vers Symfony 7.4 permet de préparer à une

potentielle montée de version vers Symfony 8.0 qui arrive en même temps. Il est plus recommandé de faire de petites mises à jour régulièrement plutôt qu'une grosse montée de version.
Aussi, suivre régulièrement les mises à jour permet de garder une réactivité en termes de support et de conseils de la communauté. Symfony étant Open-Source, il bénéficie d'une réactivité de la communauté et peut être d'une grande aide lors de debugs et de refactorisation.

Étude d'impact et évolution possible.

La montée de version a un impact clair sur toute l'application, sur les données enregistrées, ses formulaires, l'environnement de développement et même sur la méthodologie de travail.
La montée de version concerne aussi le serveur sur lequel est hébergée l'application puisqu'il faut s'assurer de la bonne version de PHP. Pour PPAP, la version de PHP est déjà supérieure à PHP 8.2 (PHP 8.4). Elle est d'office compatible avec Symfony 7.4 et Symfony 8.0.

Cette montée de version engage en particulier des zones sensibles de Symfony qui sont les formulaires; il pourrait être important de revisiter les formulaires de l'application qui sont nombreux.
De plus, l'ajout de formulaires multiples intégrés et proposé par l'ajout des FormFlow de Symfony 7.4 peut donner lieu à une amélioration et une mise à neuf complète des formulaires de PPAP qui prennent en compte : la création de devoir, le suivi des cours, le renseignement des entreprises, mais aussi la création d'utilisateurs (élèves, formateurs, tuteurs, prospects).
Aussi, l'impact sera majeur sur les dépendances et composants installés sur l'application. Certaines dépendances pourraient ne pas supporter la version 7.4.
Toutes les dépendances installées pour l'application PPAP dont il faut surveiller la compatibilité après une montée de version (hors dépendances vitales à faire fonctionner un projet symfony (doctrine, twig, maker, form, security, csrf, validator) :
- image-cropper, fixtures, doctrine-extensions, knp-snappy-bundle, authentication-bundle, cors-bundle, openai-php/symfony, phpoffice/phpspreadsheet, setasign/fpdf, symfony/discord-notifier, symfony/doctrine-messenger,

symfony/mailgun-mailer, symfony/mailjet-mailer, symfony/mercure-bundle, symfonycasts/reset-password-bundle, symfonycasts/verify-email-bundle, twig/intl-extra, twig/markdown-extra, phpunit/phpunit, symfony/phpunit-bridge, symfony/browser-kit, messenger, translation, serializer, mailer, asset.
Toutes ces dépendances doivent s'assurer une compatibilité avec Symfony 7.4 si le code vient à être monté de version.

En ce qui concerne la sécurité, il est nécessaire de faire attention à toutes les méthodes et fonctionnalités dépréciées en particulier sur les formulaires en raison de la dépréciation d'override, dans ce cas-là, la réécriture du code pourra prendre du temps. Des méthodes dépréciées sont des méthodes qui pourraient être supprimées dans le futur. En effet, les méthodes dépréciées seront supprimées en Symfony 8.0; il est donc important de se préparer à une nouvelle version.

Pour ce faire, Symfony utilise la méthode "php bin/console debug:container --deprecations". Il sera alors listé toutes les méthodes dépréciées restantes sur le code.

Analyse financière

À noter : Une montée de version sur Symfony, en particulier passer de 7.3 à 7.4 ne nécessite aucun coût en elle-même. Le coût unique reste le salaire du ou des développeurs sur le projet.

Inconvénients

Le principal inconvénient de cette montée de version reste le temps et le travail à prendre en compte.
En effet, l'adaptation du code pour sa montée de version est un travail à part entière. Des erreurs peuvent apparaître avec les méthodes dépréciées mais aussi des formulaires à réécrire. Les composants, aussi appelés 'packages', seront à surveiller. Il est alors nécessaire d'avoir une méthodologie et une documentation claires avec toutes les modifications.

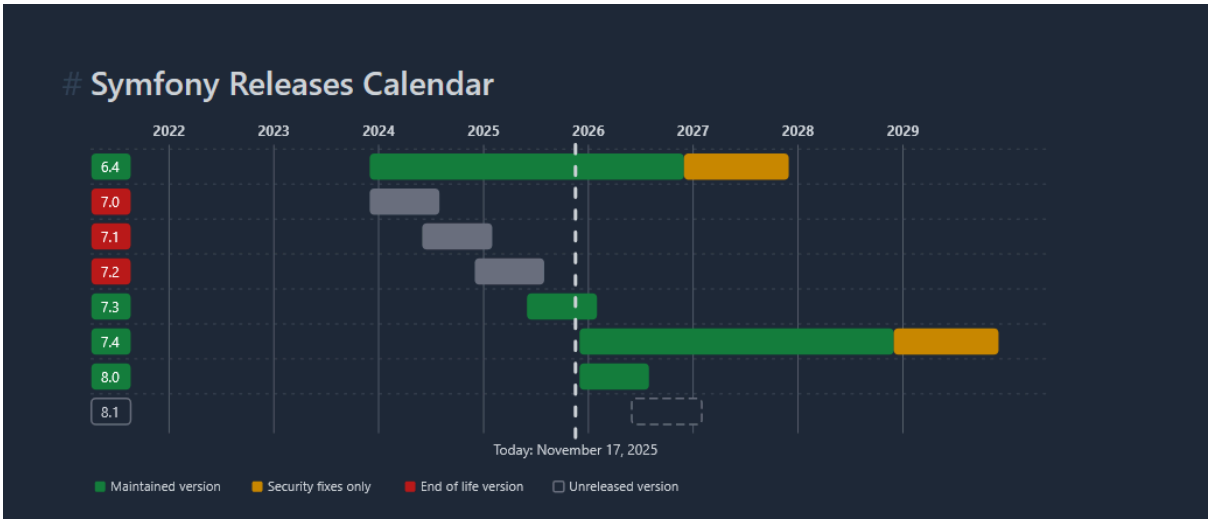
De plus, ces nouvelles avancées peuvent demander une montée en compétences (nouveaux attributs php, composants mais aussi en termes de sécurité).
En effet, la mise à jour du code mais aussi de potentielles mises à jour de packages, composants installés voire même des ajustements de configuration (en particulier avec un point de vigilance sur la dépréciation de la configuration XML avec Symfony 7.4). En clair, la montée de version peut représenter plusieurs jours de travail.

Conclusion, est-ce recommandé de faire une montée de version pour l'application PPAP?

Même s'il n'est pas encore urgent puisque des mises à jours sont encore régulières pour Symfony 7.3 (la version 7.3.9 étant sortie le 31 décembre 2025 avec des corrections de bugs de composants, amélioration de la stabilité), la montée de version régulière de Symfony, en particulier de migrer de la version 7.3 vers une version LTS comme Symfony 7.4, est recommandée afin de continuer de bénéficier d'un support régulier et d'une réactivité de la communauté. C'est un point non négligeable pour une application comme PPAP en production continue qui doit assurer un support constant entre élèves, formateurs, tuteurs et salariés. Le temps de travail nécessaire à gérer de petites mises à jour sera un gain de temps comparé au temps requis pour de grosses mises à niveau après plusieurs versions.

Des mises à jour régulières sont des changements beaucoup moins importants à absorber et elles permettent aussi de gérer son code au fur et à mesure, en particulier avec Symfony et ses méthodes dépréciées très récurrentes.
Symfony 7.4 est une passerelle de sécurité pour amener à Symfony 8.0 et la gestion des méthodes dépréciées.

L'application PPAP pourra alors évoluer avec les patchs de sécurité, toute correction de bugs et bénéficiera des améliorations continues et s'assurera de ne pas avoir de version obsolète sans support.



Pour rappel, la version 7.3 aura une fin de support en janvier 2026 tandis que la version 7.4 en LTS aura un support jusqu'en 2029.

Cela assure alors une continuité de service pour l'application et donc permet un contact constant entre élèves, formateurs, tuteurs et salariés (administrateurs, secrétariat).

Enfin, bien que l'équipe de développement doive se former aux nouvelles mises à jour, ce sera un apprentissage progressif plutôt qu'une obligation de montée en compétences de grande échelle.

Sources :

- LinkedIn: https://fr.linkedin.com/company/symfony-sas
- Feedly
- Site officiel de Symfony : https://symfony.com/ en particulier les documentations https://symfony.com/doc
- Blog de Symfony avec tous les articles à jour : https://symfony.com/blog/