



# Rapport d'avancement SAE

Hugo MONTÉ - TOAC Équipe Rouge

## Client

TOAC Triathlon – <https://toac-triathlon.com/>

## Issues principales traitées

### Ajout de Symfony (#52)

Afin de structurer le projet et prévoir sa maintenance dans le temps, l'opportunité d'utiliser un framework PHP s'est présentée. Je me suis donc chargé d'initialiser le projet Symfony, de l'ajouter au repository Git, ainsi que de configurer le projet selon nos besoins.

### Ajout de Tailwind CSS (#62)

Le site actuel utilisait du CSS sans aucune librairie, l'avantage de Tailwind CSS est sa simplicité d'usage (et d'apprentissage) ainsi que le gain de temps qu'il profère. Je me suis donc chargé de son installation et de sa configuration, que j'ai moi-même (comme mes camarades) utilisé dans les tâches qui sont des ajouts front-end.

### Interface d'administration (#56 / #58 / #59 / #61)

Sachant que nous n'utilisions pas Wordpress, la demande principale des initiaux était que l'on ajoute une interface d'administration. Je me suis occupé de son initialisation, ce qui a permis à tous mes camarades de l'utiliser, après leur avoir expliqué le fonctionnement, puis de la création de certaines parties de celle-ci :

- Gestion des articles disponibles sur le site. [#58](#)
- Gestion des utilisateurs du site (en plus de la création de l'utilisateur par défaut), tâche réalisée en collaboration avec Olivier Recher. [#58](#) / [#61](#)

Une chose à noter est que l'interface finale n'est pas la première tentative, j'avais essayé de créer tout depuis zéro, tout fonctionnait, mais pour ajouter un espace de configuration, on y passait énormément de temps à chaque fois, donc j'ai retiré tout ce travail conséquent dans l'intérêt du projet.

## **Amélioration du footer ([#66](#))**

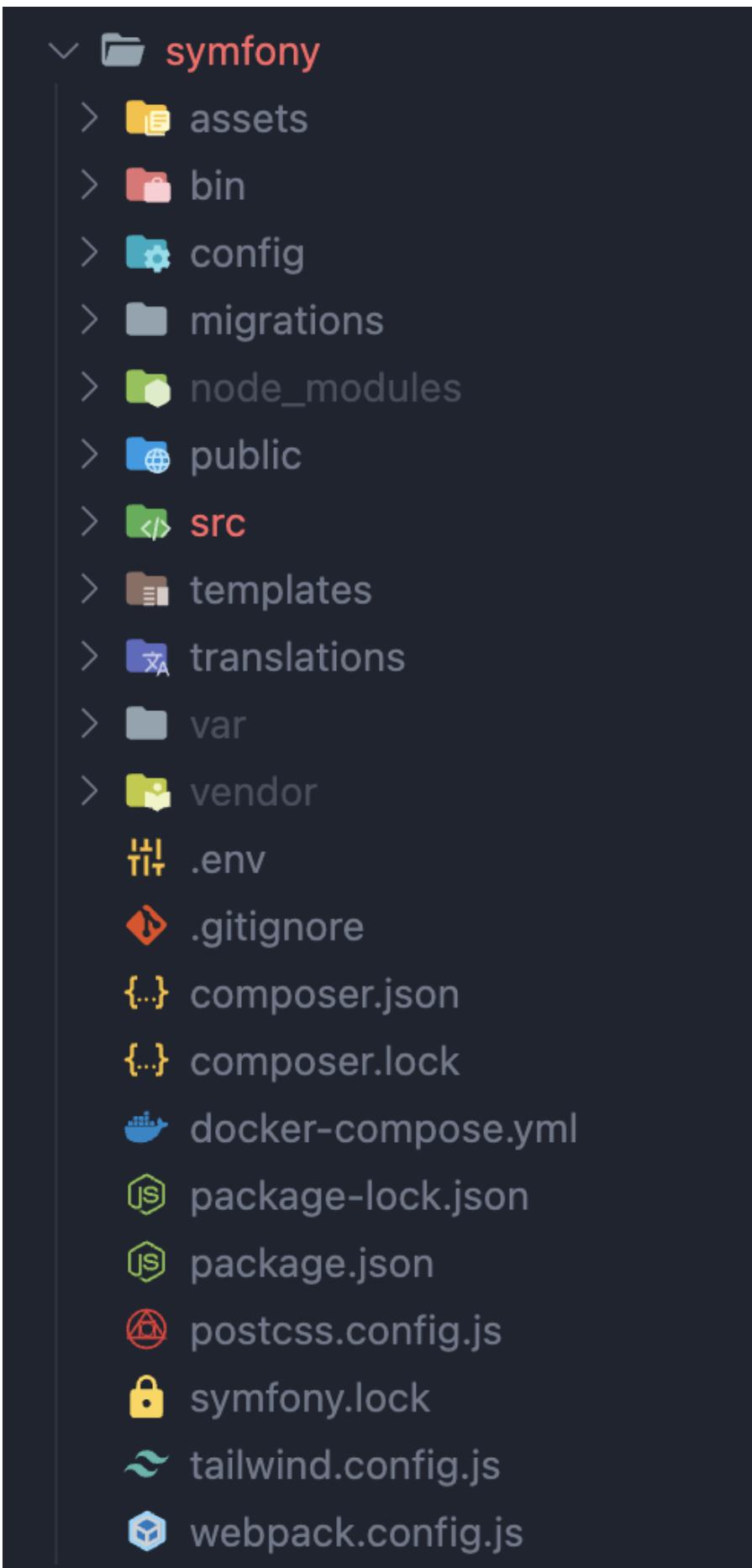
Le footer présentait des soucis visuels, je me suis chargé de corriger et améliorer son esthétique.

## **Avant / après**

Initialement, le projet que nous avons rejoint était juste une structure de fichiers HTML et CSS. À notre arrivée, nous savions qu'il allait falloir rendre ce projet plus durable. Pour ce faire, sachant que nous avions tous les quatre travaillé avec Symfony en entreprise et que ce framework correspondait à notre besoin, il a fallu initialiser Symfony. Cela ne suffisait pas, il fallait transférer tous les fichiers HTML de sorte à les intégrer. C'est une tâche que nous avons donc réalisée. Une fois ceci fait, il fallait ajouter de quoi réaliser des pages rapidement et proprement avec du CSS, c'est là où Tailwind CSS, un framework dit utilitaire s'ajoute. Grâce à cela, nous avons pu ajouter et corriger du contenu très facilement. Grâce à tout cela, nous avons réussi à améliorer la partie visible par les utilisateurs, et mis en place un espace d'administration complet, simple d'usage et surtout très extensible, avec un minimum de compétences.

## **Exemples de codes commentés**

### **Ajout de Symfony**



Voici la structure résultant d'une initialisation Symfony. On remarque aussi le fichier `tailwind.config.js` qui montre que Tailwind CSS est bel et bien installé.

## Gestion des posts

```
class PostCrudController extends AbstractCrudController
{
    public static function getEntityFqcn(): string
    {
        return Post::class;
    }

    public function configureFields(string $pageName): iterable
    {
        return [
            IdField::new('id')->hideOnForm(),
            TextField::new('title'),
            TextEditorField::new('content'),
        ];
    }
}
```

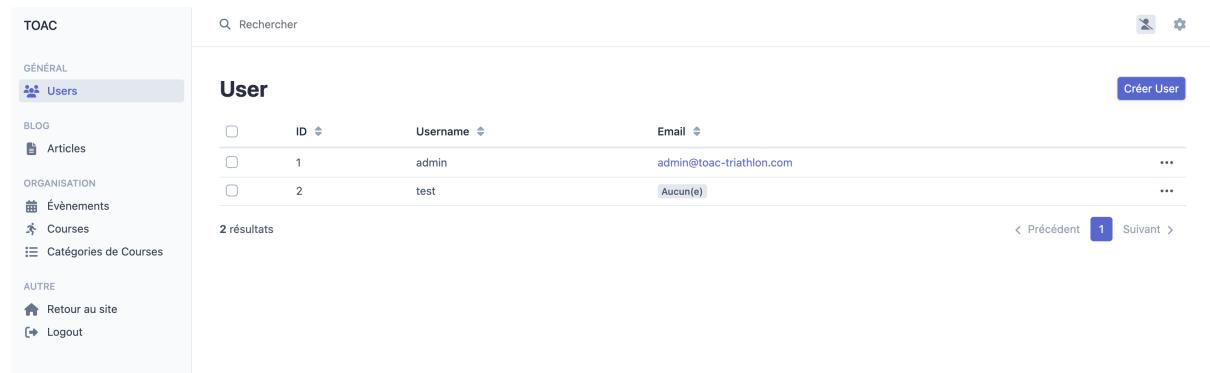
The screenshot shows the TOAC application interface. The left sidebar contains navigation links for GÉNÉRAL (Users, Articles), BLOG (Articles), ORGANISATION (Évènements, Courses, Catégories de Courses), and AUTRE (Retour au site, Logout). The main content area has a search bar and a user profile icon for 'admin'. Below it, a table lists 'Post' entries with columns for ID, Title, and Content. Each row includes a 'Voir le contenu' link and three dots for more options. At the bottom, there are navigation buttons for 'Précédent' and 'Suivant'.

ID	Title	Content
3	dzdazdza	<a href="#">Voir le contenu</a>
5	dadz	<a href="#">Voir le contenu</a>
6	dzdaz	<a href="#">Voir le contenu</a>
7	dzdazda	<a href="#">Voir le contenu</a>

The screenshot shows the TOAC application interface. The left sidebar contains navigation links for GÉNÉRAL (Users, Articles), BLOG (Articles), ORGANISATION (Évènements, Courses, Catégories de Courses), and AUTRE (Retour au site, Logout). The main content area has a search bar and a user profile icon for 'admin'. Below it, a form titled 'Modifier Post' is displayed. It has fields for 'Title\*' (with value 'dzdazdza') and 'Content\*' (with WYSIWYG editor containing 'dazadz' and 'fzadzadza'). There are two save buttons at the bottom: 'Sauvegarder et continuer l'édition' and 'Sauvegarder les modifications'.

Voici un exemple d'utilisation de l'interface [EasyAdmin](#), qui simplifie la création d'entité à configurer (ici un Post), rendant également l'interface esthétique rapidement. On voit ici l'entité liée, ainsi que les champs configurables. On notera les différents types de champ, ainsi que leur nom et leur particularité (sur `id` par exemple). Ce n'est pas la seule page que j'ai créée, mais c'est un exemple, sachant que les exemples se rejoignent visuellement.

## Gestion des utilisateurs



The screenshot shows the TOAC application's user management interface. The sidebar on the left contains links for GÉNÉRAL (Users, Articles), ORGANISATION (Événements, Courses, Catégories de Courses), and AUTRE (Retour au site, Logout). The main area has a search bar and a table titled "User". The table columns are ID, Username, and Email. It shows two results: one with ID 1, Username "admin", and Email "admin@toac-triathlon.com"; and another with ID 2, Username "test", and Email "Aucun(e)". A "Créer User" button is visible at the top right of the table. Navigation buttons for "Précédent" and "Suivant" are at the bottom right, with the page number "1" highlighted.

```
public function configureFields(string $pageName): iterable
{
    return [
        IdField::new('id')->hideOnForm(),
        TextField::new('username'),
        EmailField::new('email'),
        TextField::new('password')
            ->setFormType(PasswordType::class)
            ->setRequired($pageName === Crud::PAGE_NEW)
            ->onlyOnForms(),
    ];
}
```

```

private function hashPassword() {
    return function($event) {
        $form = $event->getForm();
        if (!$form->isValid()) {
            return;
        }

        if (!$password = $form->get('password')->getData())) {
            return;
        }

        $hash = $this->userPasswordEncoder->hashPassword($this->getUser(), $password);
        $form->getData()->setPassword($hash);
    };
}

```

On voit ici un exemple similaire au précédent, sauf que cette fois-ci, il fallait gérer le hash (différent du chiffrement, car irréversible) des mots de passe.

## Les risques / les freins

Le souci d'arriver dans un projet déjà réalisé peut varier en termes de sévérité. Parfois le projet est tellement vaste qu'il n'est pas possible de repartir de zéro ou totalement migrer. Dans notre cas, les fichiers existants étaient assez aisément ajoutables à Symfony, car il y en avait que quelques dizaines. Une contrainte supplémentaire liée à cela, était qu'après avoir migré vers Symfony, les initiaux ont continué à modifier les fichiers d'avant la migration, ce qui ralentissait notre travail. Ajoutés à cela, les créneaux entre les initiaux et les alternant différaient, ce qui rendait notre collaboration moins simple. Cela représente les freins principaux, mais pas l'ensemble de ceux présents.

## Conseils pour la suite

Pour mes camarades en formation initiale, le conseil que j'aurais à donner serait de continuer à se former à Symfony, et monter en compétences, tout en privilégiant l'intérêt du client. Il est nécessaire de se former aux technologies modernes afin de ne pas être mis de côté sur le marché du travail, une fois qu'il faudra faire face à la concurrence. Dans le cadre de ce projet, il serait de le mener à bien, en ne cessant pas d'améliorer les interfaces ainsi que les fonctionnalités.

## Note de fin

Ce document ne démontre peut-être pas assez ce qui a pu être fait durant ces deux semaines, en revanche le code parlerait de lui-même (même si plus long à regarder), et ce qui a été fait est fonctionnel, et je l'espère sera encore plus amélioré par les initiaux qui continuent le projet, et que le client sera satisfait.