DEPARTEMENT INFORMATIQUE
INSTITUT UNIVERSITAIRE DE BLAGNAC
UNIVERSITE TOULOUSE II – JEAN JAURÈS

SAE - TOAC

Recher Olivier
3B

Destinataire :

M. Bruel

15/10/2023

Sommaire

Issues réalisées lors du projet

Pendant le projet de réalisation d'un site vitrine pour le TOAC, j'ai eu l'opportunité de travailler sur diverses tâches essentielles pour assurer le bon fonctionnement et la qualité du site. Ces tâches étaient cruciales pour atteindre les objectifs du projet et offrir une expérience utilisateur optimale. Je me suis principalement concentré sur la sécurité de la connexion au tableau de bord administrateur, la gestion des utilisateurs ainsi que des corrections dans l'aspect du site, notamment l'ajout d'une barre de navigation plus ergonomique. Voici un aperçu des principales tâches que j'ai traitées.

Mise en place de la sécurité à la connexion de la page admin (#57)

Au début de cette phase du projet, ma première responsabilité a été de renforcer la sécurité de l'interface d'administration en travaillant sur la mise en place de mesures de sécurité. La protection de l'accès à cette zone sensible était une priorité, et j'ai mis en œuvre des protocoles de sécurité robustes pour empêcher tout accès non autorisé en utilisant les outils offerts par Symfony.
Pour accomplir cette tâche, j'ai utilisé le package Security de Symfony, qui permet de gérer efficacement la sécurité de l'application. Ci-dessous, vous trouverez une capture d'écran du fichier security.yaml, qui permet de configurer le comportement de l'application en cas de tentative de connexion.

```yaml
security:
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                # Make link with User entity for authentication
                class: App\Entity\User
                property: username
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider
            # activate different ways to authenticate
            # define how your users will authenticate
            form_login:      You, last week • feat(dashboard) set up firewall, redirection
                login_path: app_login
                check_path: app_login
            logout:
            # where to redirect after logout
                path: /logout
                target: homepage
```

La partie ci-dessous permet de sécuriser une partie de site en limitant l'accès qu'à certains utilisateurs. Ici les administrateurs pour tous les chemins commençant par "/admin".

```yaml
access_control:
# Securise l'accès à la page d'administration en redirigeant vers la page de connexion
    - { path: ^/admin, roles: ROLE_ADMIN }
```

Si l’utilisateur qui cherche à se connecter n’est pas authentifié ou n’est pas administrateur, le firewall bloque l’accès et renvoie vers le chemin d’authentification, ici “app_login”.

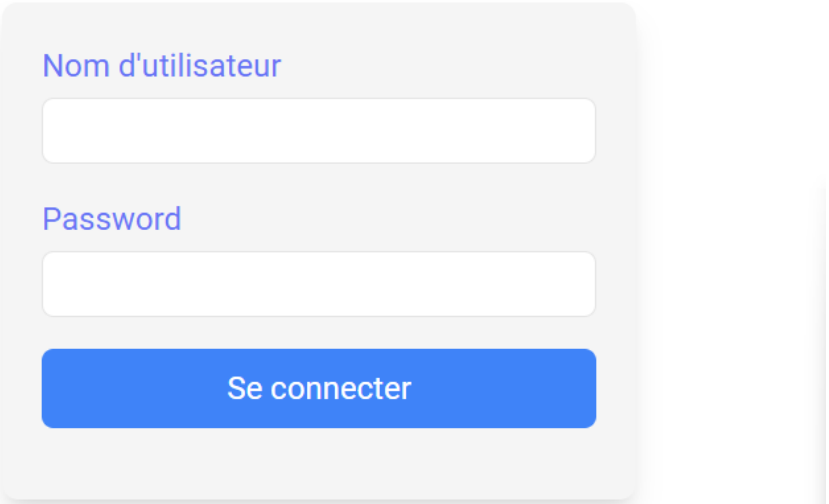
Page de connexion admin (#60)

Une fois la sécurité mise en place, l'objectif était de mettre en place une gestion efficace de la connexion pour les administrateurs du site. Cette étape revêtait une importance cruciale pour garantir que l'équipe de gestion puisse accéder en toute sécurité à l'interface d'administration du site, en vue de gérer et de maintenir le contenu. Pour atteindre cet objectif, j'ai continué à utiliser les outils de Symfony, en créant notamment un "LoginController" dédié, permettant de définir le comportement lors d'une tentative d'authentification.

```php
class LoginController extends AbstractController
{
    // définition de la route de la page de connexion
    #[Route('/login', name: 'app_login')]
    public function index(AuthenticationUtils $authenticationUtils): Response
    {
        // AuthenticationUtils est l'object symfony gérant la connexion
        // validant les identifiants et les mots de passe

        // Récupération de l'erreur de connexion
        $error = $authenticationUtils->getLastAuthenticationError();
        // Récupération du dernier nom d'utilisateur saisi (si il y en a un)
        $lastUsername = $authenticationUtils->getLastUsername();      You, last week

        // Charge la page de connexion en lui passant les variables
        return $this->render('login/index.html.twig', [
            'controller_name' => 'LoginController',
            'last_username' => $lastUsername,
            'error'         => $error,
        ]);
    }
}
```

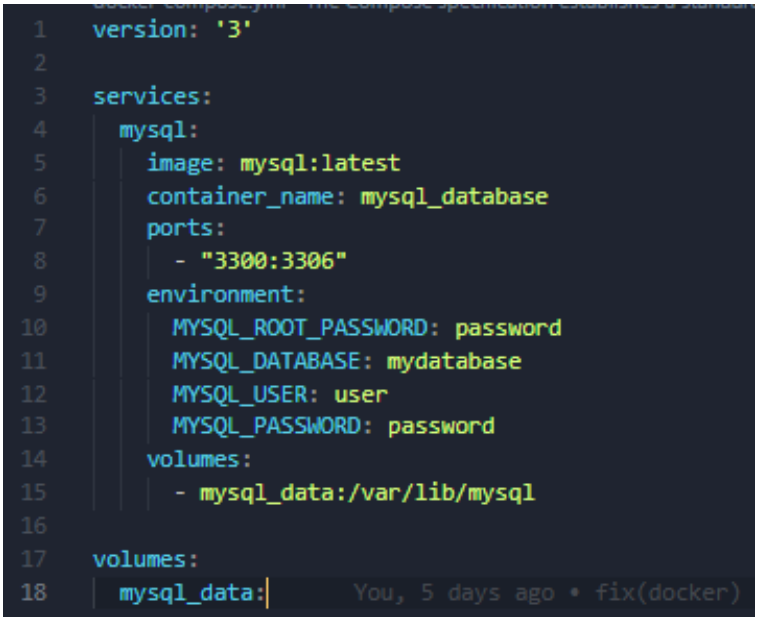
Ce contrôleur renvoie la page de connexion ou, en cas de validation du formulaire, redirige vers le tableau de bord. J'ai également consacré du temps à améliorer l'interface utilisateur de l'application, dans le but d'offrir un affichage soigné et efficace pour les utilisateurs. Ci-dessous, vous trouverez le code ainsi que le rendu de la page de connexion.

```twig
{# extend du layout sans la barre de navigation et le footer #}
{% extends 'layouts/default/empty.html.twig' %}
{% block body %}
    {# ... #}
    {# Affichage des erreurs de connexion #}
    {% if error %}      You, last week • feat(dashboard) set up firewall, redirection to /…
        <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
    {% endif %}

    <div class="flex h-screen bg-white">
    {# Prise en compte du format mobile pour adapter la largeur de la box à l'écran #}
        <div class="sm:w-1/4 w-full sm:max-w-xs m-auto bg-neutral-100 rounded-lg shadow-lg p-5">
          {# Redirection au Controller une fois envoyer #}
        <form action="{{ path('app_login') }}" method="post">
            <div class="space-y-1">
                <label class="text-indigo-500" for="username">Nom d'utilisateur</label>
                {# Affiche le dernier nom saisi renvoyé par le controller #}
                <input class="w-full" type="text" id="username" name="_username" value="{{ last_username }}">
            </div>
            <div class="space-y-1">
                <label class="text-indigo-500" for="password">Password</label>
                <input class="w-full"  type="password" id="password" name="_password">
            </div>
            <div>
                <button class="button-primary" type="submit">Se connecter</button>
            </div>
            {# Permet de rediriger la sortie en cas de validation du formulaire #}
        <input type="hidden" name="_target_path" value="/admin">
```



Correction Docker (#55)

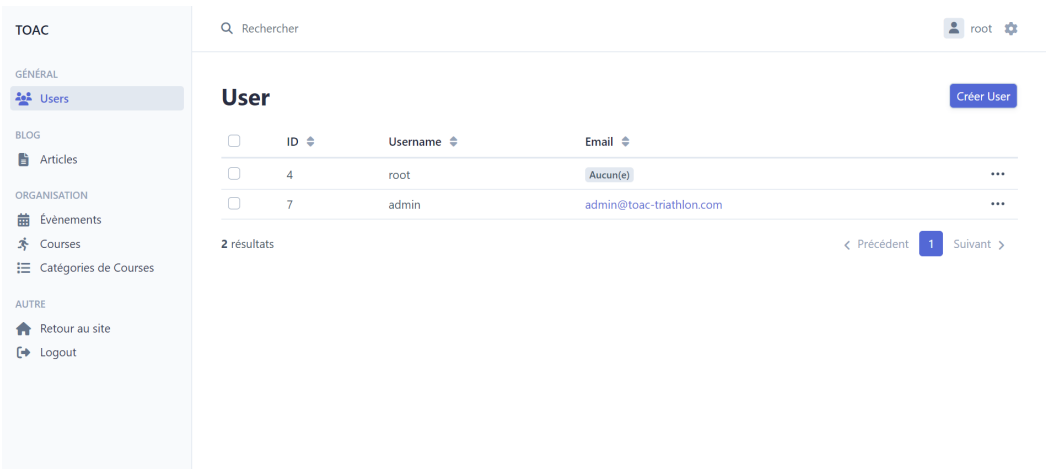
Un peu plus tard dans le projet, j'ai résolu des problèmes liés à Docker. Docker nous permet de nous connecter à une base de donnée local nécessaire pour le bon fonctionnement du tableau de bord administrateur. Une erreur survenait chez certain membre de l'équipe où la connexion avec la base de donnée ne se faisait pas correctement. Ayant déjà travaillé avec Thomas sur la mise en place du docker, nous avons vu ensemble comment corriger ce point. L'erreur provenait d'un fichier qui réécrivait par-dessus le fichier de configuration docker-compose.yml.

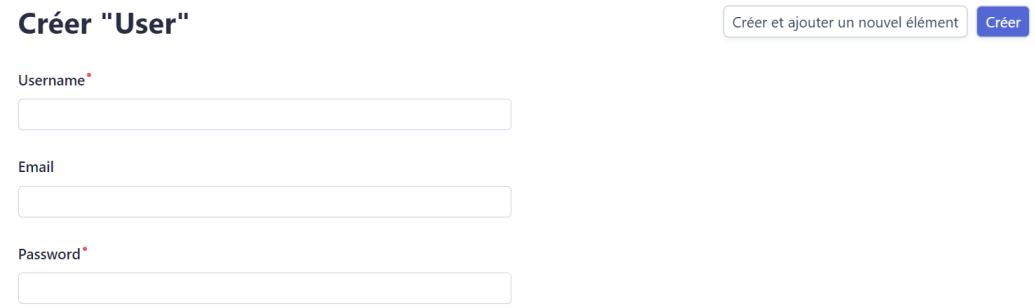


Espace création d'utilisateur (#61)

Après avoir pris en charge l'accès au tableau de bord, la prochaine étape logique consistait à mettre en place un espace de gestion des utilisateurs, permettant de créer de nouveaux comptes administrateurs, de les supprimer ou de les modifier. Cette fonctionnalité a été réalisée en exploitant les fonctionnalités offertes par Symfony, en particulier EasyAdmin, notamment grâce à la commande suivante : "make:admin:crud". Le contrôleur CRUD prend en charge de manière automatisée l'ensemble des phases que l'entité utilisateur peut traverser (create, render, update, delete). Voici ce qui a été créé, permettant ainsi la gestion de différents formulaires et le rendu associé.

```php
public function createNewFormBuilder(EntityDto $entityDto, KeyValueStore $formOptions, AdminContext $context): FormBuilderInterface
{
    $formBuilder = parent::createNewFormBuilder($entityDto, $formOptions, $context);
    return $this->addPasswordEventListener($formBuilder);
}

public function createEditFormBuilder(EntityDto $entityDto, KeyValueStore $formOptions, AdminContext $context): FormBuilderInterface
{
    $formBuilder = parent::createEditFormBuilder($entityDto, $formOptions, $context);
    return $this->addPasswordEventListener($formBuilder);
}

private function addPasswordEventListener(FormBuilderInterface $formBuilder): FormBuilderInterface
{
    return $formBuilder->addEventListener(FormEvents::POST_SUBMIT, $this->hashPassword());
}
```





Édition du mot de passe d'un utilisateur (#60)

Avec Hugo, nous avons travaillé sur la mise en place de la possibilité pour les utilisateurs de modifier leur mot de passe à partir du menu d'édition de leur profil. Cela a nécessité la recherche et l'implémentation des outils appropriés dans Symfony pour le hachage sécurisé des mots de passe et leur conservation en base de données. Étant donné que la sécurité est une priorité absolue pour une fonctionnalité de ce type, nous avons veillé à sa mise en place de manière fiable.

Nouvelle barre de navigation (#64)

J'ai également résolu un problème lié à la barre de navigation du site. Une navigation fluide et intuitive est essentielle pour offrir une expérience utilisateur optimale, et j'ai veillé à ce que cette composante cruciale fonctionne de manière correcte. En effet, j'ai apporté des améliorations significatives à la barre de navigation en corrigeant l'affichage du sous-menu lors du survol, même en dehors de la barre de navigation. Pour ce faire, j'ai opté pour l'utilisation du composant Flowbite, qui est déjà compatible avec tailwindcss. Cette solution s'est avérée être la plus adaptée à notre projet tout en permettant de gagner du temps de développement.

```
{# Ajour d'une barre de navigation #}
<nav class="bg-white border-gray-200">
  <div class="max-w-screen-xl flex flex-wrap items-center justify-end mx-auto p-4">
    <div class="hidden w-full md:block md:w-auto" id="navbar-multi-level">
      <ul class="flex flex-col font-medium p-4 md:p-0 mt-4 border border-gray-100 rounded-lg bg-gray-50 md:flex-row md:space-x-8 md:
       <a>
        <li>
          <a href="/" class="block px-4 py-2 text-gray-700 hover:text-blue-700 md:p-0 absolute left-0">
          {# chemin dynamique vers le logo du TOAC #}
            <img src="{{  asset('assets/index_top.jpg') }}" alt="half du Lauragais" class="w-full h-6">
          </a>
        </li>
       <li>
          <button id="dropdownNavbarLink1" data-dropdown-toggle="dropdownNavbar1" class="flex items-center justify-between w-full py-2
            <svg class="w-2.5 h-2.5 ml-2.5" aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 10 6">
              <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="m1 1 4 4 4-4"/>
            </svg>
          </button>
          <!-- Dropdown menu -->
          <div id="dropdownNavbar1" class="z-10 hidden font-normal bg-white divide-y divide-gray-100 rounded-lg shadow w-44 ">
              <ul class="py-2 text-sm text-gray-700" aria-labelledby="dropdownLargeButton">
                <li>
                {# chemin dynamique vers la page progammation du TOAC #}
                  <a href="{{ path('schedule') }}" class="block px-4 py-2 hover:bg-gray-100">Programme</a>
```

Il a également été nécessaire d'ajouter des références vers les fichiers app.js et app.css, permettant respectivement d'importer le script de fonctionnement de la barre de navigation Flowbite et la personnalisation du composant tailwindcss.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Triathlon de Nailloux | Half du Lauragais</title>
    <link rel="stylesheet" href="{{ asset('style/style.css') }}">
    <link rel="stylesheet" href="{{ asset('style/footer_style.css') }}">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    {# Importation du fichier style config tailwind #}      You, 1 second ago
    {{ encore_entry_link_tags('app') }}
    {# Importation du script pour fonctionnement de la barre de navigation #}
    {{ encore_entry_script_tags('app') }}
```

```
import './styles/app.css';
import 'flowbite';      You, yesterday • feat(navbar)
```

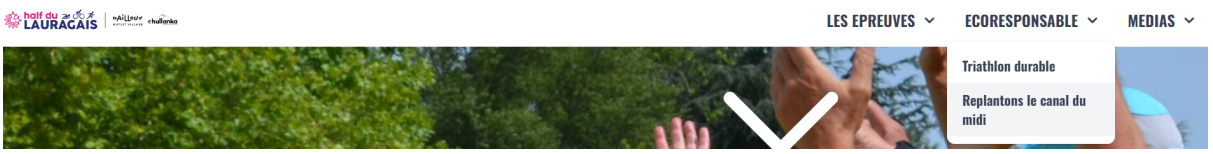
app.js

```
You, 6 days ago | 2 authors (hugomonte and others)
@tailwind base;
@tailwind components;
@tailwind utilities;

body {
    background-color: lightgray;
}

.button-primary {
    @apply bg-blue-500 text-white py-2 px-4 rounded-md hover:bg-blue-600 transition-all;
}
```

app.css

Avant :

Après :

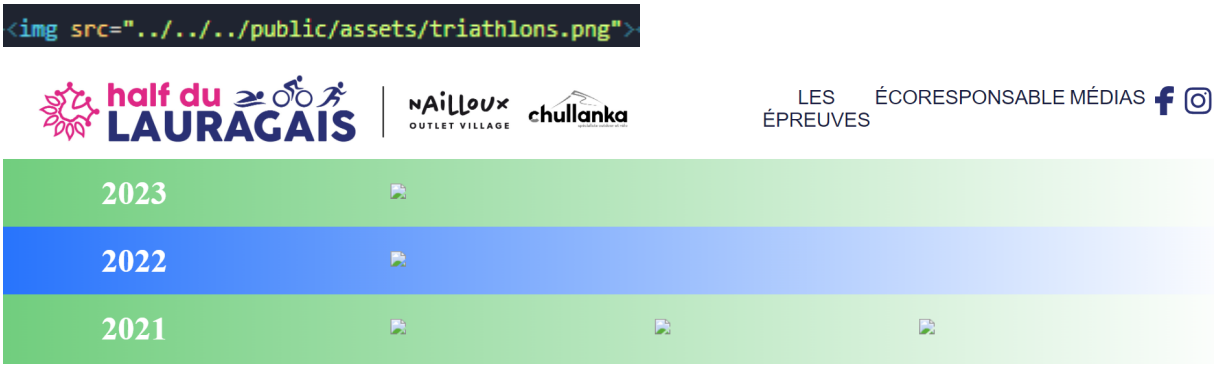


Correction de régression des pages (#67)

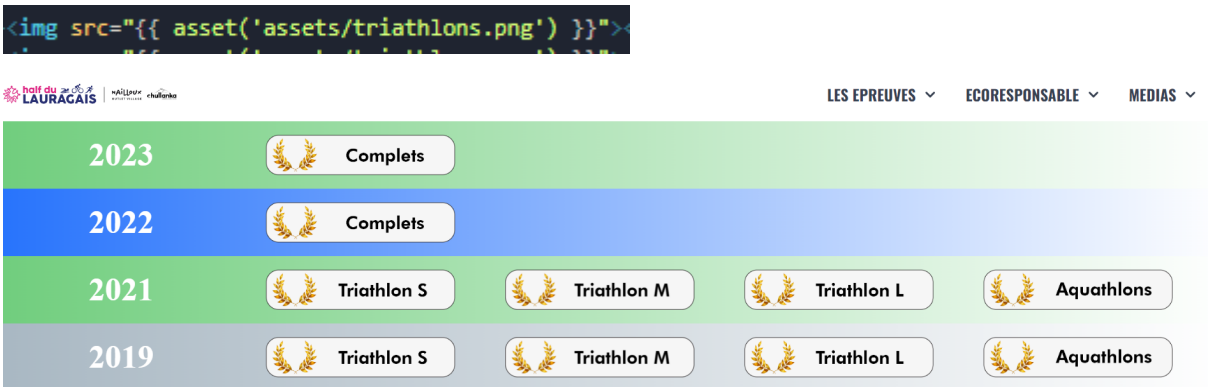
Nous, le groupe d'alternants, avions déjà travaillé lors de la migration du projet vers Symfony, à adapter le travail qui avait déjà été réalisé par les initiaux afin de le rendre cohérent avec le langage Twig de Symfony. Nous avions mis en place des templates, des variables ainsi que des getters afin de récupérer de manière propre les différentes ressources nécessaires à la page internet. Cela incluait notamment les liens vers les fichiers de style et les images, qui n'utilisaient plus des chemins relatifs, mais des chemins dynamiques vers le dossier "/public". Cependant, en raison d'un manque de communication et de compréhension avec les initiateurs, ce travail a été écrasé par leurs modifications, ce qui a nécessité de réitérer cette étape, plusieurs fois.

Voici un exemple de correction de chemins en dur en faveur de l'utilisation de chemins dynamiques :

Avant :



Après :



En plus de cela, il y avait des points à corriger dans le style du site afin de correspondre au projet. En effet, j'ai rencontré une situation où une classe "hidden" était configurée dans les fichiers CSS, alors que "hidden" était déjà utilisé dans tailwindcss. Ce détail en apparence mineur a entraîné un conflit, provoquant des comportements inattendus en termes de style. Pour remédier à cela, j'ai modifié la classe "hidden" en "invisible" dans l'ensemble des fichiers.

```css
.hidden {
    visibility:hidden ;
}
```

```css
.invisible {
    visibility:hidden ;
}
```

Les risques/Freins

J’identifierai comme principal frein que l’on a pu rencontrer, le manque de temps que nous avons eu avec les initiaux pour travailler ensemble sur le projet. En effet, nous ne les avons rencontrés qu'à la première séance, ce qui a compliqué la communication relative à nos tâches respectives. Malgré ces défis, nous avons réussi à progresser. Cependant, nous nous sommes retrouvés dans une situation où nous travaillions sur la dernière version du projet, alors que les membres initiaux travaillaient sur des fichiers HTML devenus obsolètes. Cette situation a conduit à de nombreuses régressions en termes de style et de chemins d'accès aux ressources lors de la fusion. Une communication plus efficace aurait pu nous faire gagner du temps et favoriser une meilleure collaboration en vue d'une avancée harmonieuse.

Conseil pour la suite

Pour la suite, il sera important pour les membres du groupe de s’adapter aux changements que nous avons pu apporter à la structure pour permettre une cohérence dans le projet. Nous avons pris le temps de bien leur expliquer afin qu’ils ne rencontrent plus de soucis par la suite. Afin d’assurer un rendu professionnel au client, il sera nécessaire d’ajuster quelques détails dans l’aspect du site afin d’offrir un produit fonctionnel et agréable pour le client.

Conclusion

Au cours de ce projet de développement du site du TOAC, j'ai eu l'opportunité de travailler sur des aspects cruciaux tels que la sécurité, la gestion des utilisateurs et l'amélioration de l'interface utilisateur. La découverte d'EasyAdmin s'est révélée être un atout majeur, simplifiant considérablement la gestion de l'administration. Cependant, des défis sont apparus en raison du manque de communication avec l'équipe des initiaux, ce qui a entraîné des régressions dans le projet. Néanmoins, ces difficultés ont renforcé notre capacité d'adaptation et d'apprentissage.