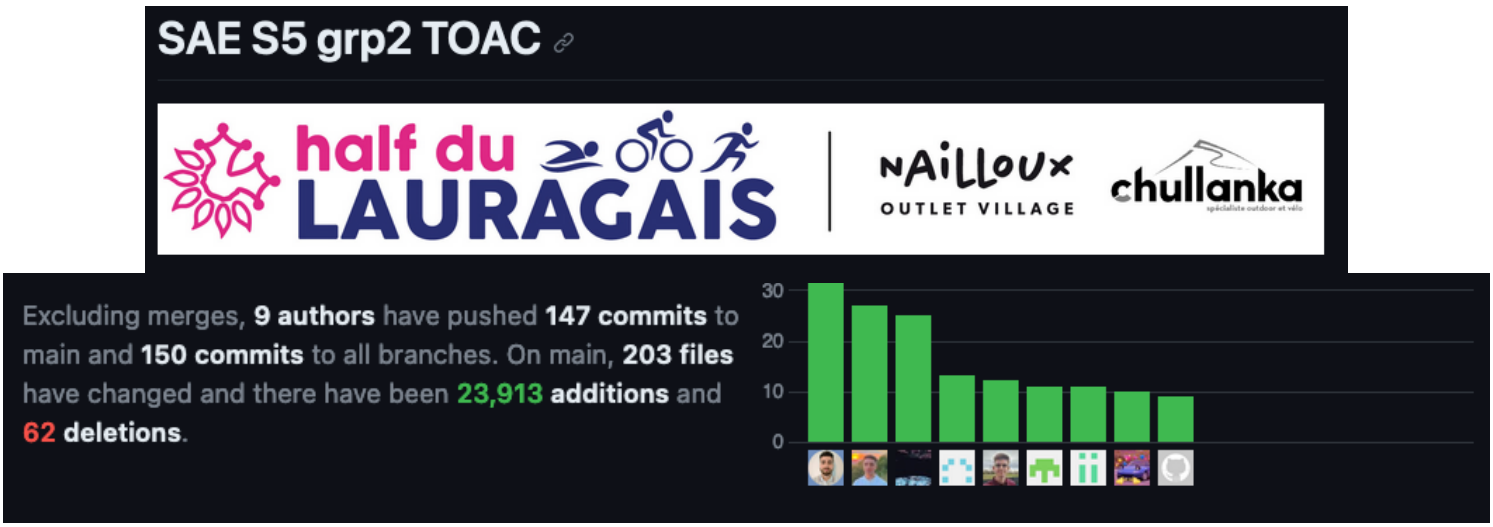
TOAC Groupe 2

Rapport d’Avancement de Projet



Graphique des contributeurs du projet sur GitHub

J’ai été intégré sur le projet de la refonte du site web du TOAC de , refait initialement en HTML CSS et Javascript.

Voici les ajouts que j’ai pu apporté au projet sur les deux semaines qui y ont été consacrées.

User Story Traitée

La User story générale qui a été traitée par l'ensemble du groupe est l'ajout de nouvelles informations par un *dashboard* administrateur. Pour faire simple, il faut que lorsqu'un administrateur veut changer le contenu du site (prix des courses, nouvelles actualités etc…), il puisse le faire sans avoir besoin de modifier le code HTML.

Pour cela, nous avons dû mettre en place une technologie *backend*, qui permet de se connecter, et de modifier des informations qui seraient stockées en base de données. Nous avons remarqué que nous [alternants] faisions tous du Symfony (*framework backend PHP*), et que son utilisation paraissait justifiée ici sans que la mise en place ne nous monopolise tout le temps alloué.

J'ai travaillé en particulier sur l'ajout des courses dans le dashboard, permettant aux administrateurs de créer de nouvelles courses, d'indiquer leurs prix, de donner des détails sur le parcours etc…

Avant / Après

Comme dit précédemment, le site avait été développé par les initiaux en utilisant uniquement du HTML/CSS/JS, ce qui ne semblait pas optimal pour résoudre le problème de *dashboard* administrateur dans les temps.

Les fonctionnalités qui leurs manquaient étaient le panneau de configuration des administrateurs, et quelques pages qui n'étaient pas terminées ou auxquelles il manquait du style.

Le problème qui nous a semblé le plus urgent de résoudre a été le panneau de configuration. C'est pourquoi nous avons commencer par installer Symfony, framework backend PHP, avec lequel nous pourrions gérer les connexions des administrateurs et la création de nouveaux posts, articles, courses etc…

Le premier jour, nous avions donc intégré symfony au projet, et avions commencé par reprendre toutes les pages HTML développées par les initiaux et les transformer en *templates* symfony Twig. Le Twig est un *superset* de l'HTML, c'est à dire que le code HTML est valide en Twig, mais que des fonctionnalités en plus sont disponibles, notamment le découpage par bloc,

qui ont permis de découper les Navbar et Footers en pages à part, afin que les modifications de ces derniers soient reportées sur l'ensemble du site.

Une fois Symfony ajouté au projet, je me suis occupé personnellement de rajouté un Docker afin d'avoir une base de données MariaDB pour stocker les entités Symfony.

Code commenté

Voici le code que j'ai écrit pour gérer les courses dans le dashboard administrateur:

```php
class CourseCrudController extends AbstractCrudController
{
    1 usage
    private $em;

    Thomas MASIN (SAN DE RGOCC)
    public function __construct(EntityManagerInterface $em)
    {
        $this->em = $em;
    }

    hugomonte
    public static function getEntityFqcn(): string
    {
        return Course::class;
    }

    Thomas MASIN (SAN DE RGOCC) +1
    public function configureFields(string $pageName): iterable
    {
        return [
            TextField::new( propertyName: 'titre', label: 'Titre'),
            TextField::new( propertyName: 'format', label: 'Format'),
            NumberField::new( propertyName: 'prix', label: 'Prix'),
            TextField::new( propertyName: 'challenge', label: 'Challenge'),
            AssociationField::new( propertyName: 'courseCategory', label: 'Catégorie')
        ];
    }
}
```

Symfony/src/controller/Admin/CourseCrudController.php

Ce code gère le formulaire de création et de modification des Courses, une entité Symfony (classe) que j'ai également créé. On voit ici l'avantage de EasyAdmin, la librairie utilisée pour le dashboard, qui permet de créer un

```php
#[ORM\Entity(repositoryClass: CourseRepository::class)]
class Course
{
    1 usage
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    2 usages
    #[ORM\Column(length: 255)]
    private ?string $titre = null;

    2 usages
    #[ORM\Column(length: 255)]
    private ?string $format = null;

    2 usages
    #[ORM\Column]
    private ?int $prix = null;

    2 usages
    #[ORM\Column(length: 255, nullable: true)]
    private ?string $challenge = null;

    2 usages
    #[ORM\ManyToOne(inversedBy: 'courses')]
    #[ORM\JoinColumn(nullable: false)]
    private ?CourseCategory $courseCategory = null;
```

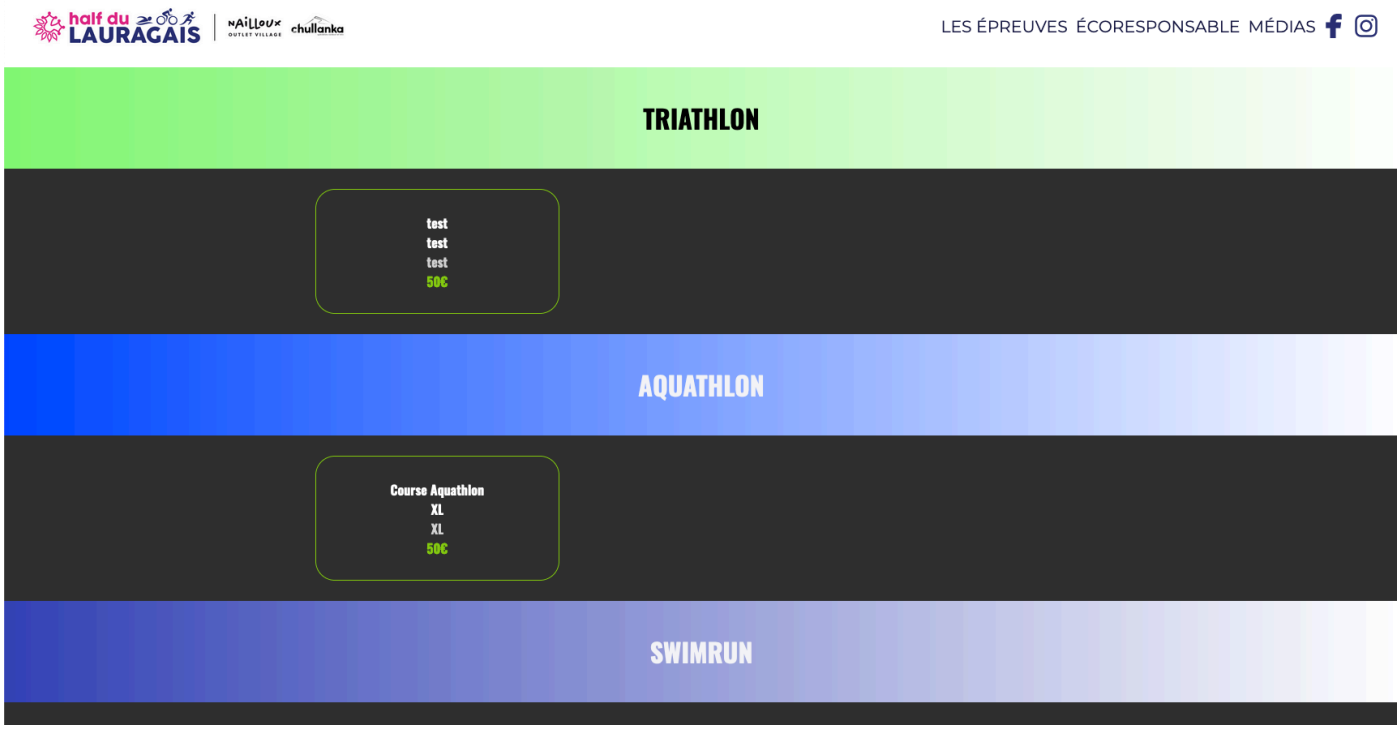
Symfony/src/entity/Course.php

formulaire rapidement.

Le code ci contre est la définition de l'entité symfony, avec tous les champs qui lui sont propres.

Par exemple, le format de la course est une chaine de caractère de longueur maximale 255.

Il y a également une catégorie de course, liée à une autre entité que j'ai créé, CourseCategory, dont je montre l'utilité ci-dessous.

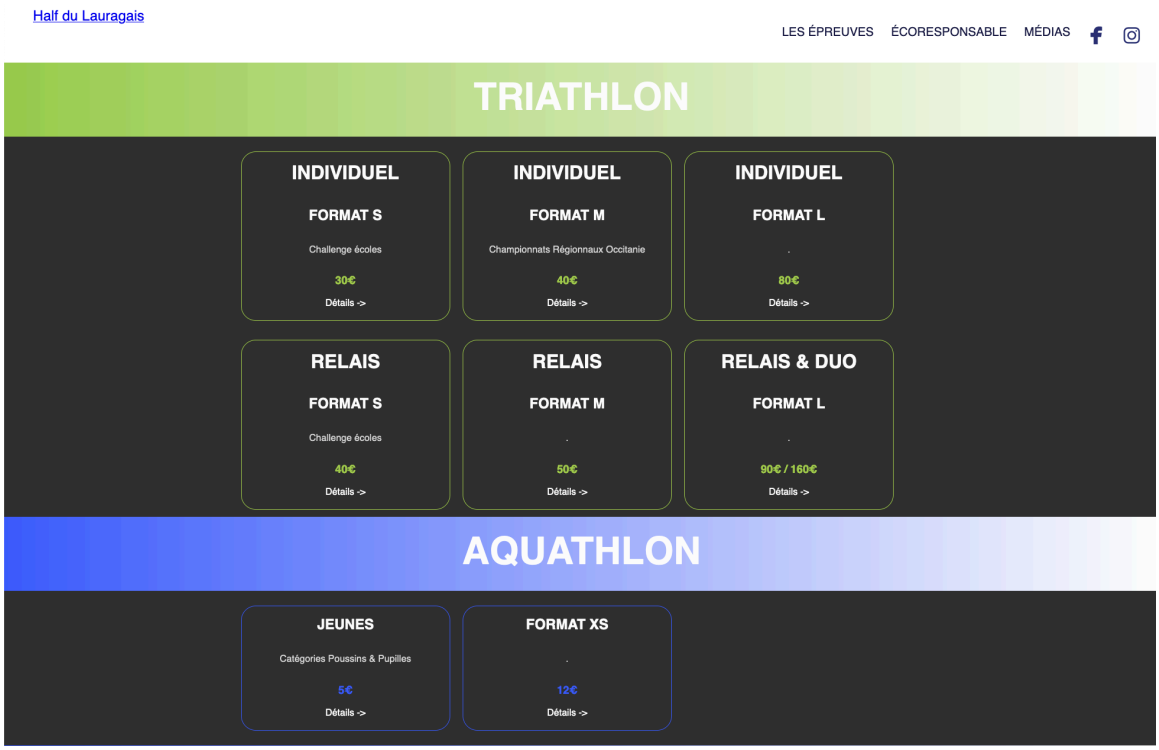


Ici, la page de course, avec un bandeau de couleur pour le nom de la catégorie, et en dessous, toutes les courses liées à cette catégorie.

La couleur et le nom de la catégorie est modifiable, et tous les attributs des courses sont modifiables.

La seule fonctionnalité manquante ici est l'affichage de la course en détail. Cet ajout devrait prendre environ une demi-journée à une journée.

Voici l'ancienne page des courses:



Peu de différences esthétiques, le seul changement et la possibilité de modification par un administrateur sans toucher au code.

Ici, on a une page statique HTML, contre une page Twig qui permet de gérer les courses automatiquement. Voici le code Twig.

```twig
{% extends 'layouts/default/base.html.twig' %}
{% block body %}
    <link rel="stylesheet" href="{{ asset('style/courses_style.css') }}">
    <div class="main-container">
        {% for category in categories %}
            <div class="section">
                <div class="section-title"
                style="...">
                    {% if category.colour[1:3] > 'cc' or category.colour[3:5] > 'cc' %}
                        <h1 class="text-black">{{ category.name }}</h1>
                        {% else %}
                        <h1 class="text-gray-100">{{ category.name }}</h1>
                    {% endif %}
                </div>
                <div class="section-content">
                    {% for race in category.getCourses %}
                        <a href="{{ path('app_race_show', {'id' : race.id}) }}"
                           class="w-max-1/3 p-7 text-center hover:bg-amber-400 w-full border border
                            <h1>{{ race.titre }}</h1>
                            <h2>{{ race.format }}</h2>
                            <p style="...">{{race.format}}</p>
                            <h3 class="price text-lime-500">{{ race.prix }}€</h3>
                        </a>
                    {% endfor %}
                </div>
            </div>
        {% endfor %}
```

Symfony/templates/race/index.html.twig

Les risques / Les freins

L'un des principaux freins est la contrainte de temps. En deux semaines, il est compliqué d'ajouter une nouvelle technologie a un projet et de former l'équipe existante à cette technologie.

Le risque qui en découle est que l'équipe d'initiaux n'est pas compris ce que nous avons fait, et donc qu'ils n'arrivent pas à poursuivre le travail en Symfony. Pour cela, je me rends joignable même après la période à l'IUT pour répondre à leurs éventuelles questions et à les aider si besoin.

Un autre risque est que le groupe n'ait pas le temps de reprendre tout le site en symfony et en HTML/CSS/JS avant la fin du temps imparti, là où l'autre groupe TOAC, qui a repris le site en WordPress, technologie déjà utilisée par le site actuellement, aura probablement fini plus tôt avec un résultat similaire.

Enfin, le dernier frein a été la difficulté de compréhension avec le client. Le client, qui a demandé la nouvelle version du site, n'a pas fourni toutes les informations aux élèves initiaux du groupe, et chaque nouvelle question entrainait un délais de réponse trop long pour que les réponses soient utilisables.

Conseils pour la suite

Mon conseil aux initiaux pour le temps restant sur cette SAÉ est de se concentrer sur le visuel du site. Un site beau graphiquement, et semblant robuste et attractif, auquel il manque quelques fonctionnalités mineures, aura toujours plus de chances d'être apprécié par le client qu'un site fonctionnel à 100% auquel il manquerait des éléments importants de style.

Je rappelle également que je suis disponible sur Discord, par mail ou par téléphone en cas de question ou de problèmes liés à Symfony ou à Docker pour les initiaux jusqu'à la fin de la SAÉ.