

L'association **PAUL** **Green**

2022



Dossier PROJET

Table des matières

**Table des matières**



Avant-Propos

Ce dossier professionnel a été réalisé dans le cadre d’un projet de remise à niveau professionnel suite à un licenciement économique. Informaticien de métier, lors de mes premiers entretiens avec ma conseillère Pôle emploi j’ai évoqué l’idée de suivre une formation en développement informatique vers un secteur porteur. C’est ainsi qu’elle a pu me proposer de suivre une formation de Développeur Web et Web mobile en ayant l’opportunité d’obtenir un financement auprès de Pôle emploi. J’ai donc pris contact avec l’organisme de formation ARINFO de Saint Nazaire.

J’ai tout de suité été séduit par le parcours proposé par l’organisme, le concept « Learning by doing » me permettant de toujours découvrir et apprendre, et assouvir la curiosité qui me caractérise.

J’ai également été attiré **par la variété des emplois** **et des thèmes** que regroupe le métier de développeur Web, et par la **possibilité d’évolution** qu’il propose dans une carrière.

En effet, je suis informaticien de métier d’abord dans un premier temps en tant que développeur informatique exercé pendant plus de 20 ans puis en tant que technicien de maintenance en informatique pendant 9 ans.

J’ai choisi cette formation car j’ai voulu revenir à mon métier de base et actualiser mes compétences en matière de développement en me familiarisant avec les outils du web pour concevoir des sites et découvrir de nouveaux langages de programmation web afin d’être plus en phase avec la réalité du marché du travail.

Cette formation a duré 6 mois, du 2 novembre 2021 au 30 avril 2022.

**Remerciements**

Je tiens à remercier Nicolas GICQUEL notre formateur, pour la qualité de sa formation, sa disponibilité, sa pédagogie. Elodie BOINET directrice du centre ARINFO de Saint-Nazaire pour son soutien et ses conseils pour réaliser cette formation, ainsi que Magali VINCE qui m’a donné des conseils quant à la présentation et la rédaction de mon CV.

Je remercie également ma conseillère Pôle Emploi pour m’avoir soumis cette proposition et suivi mon dossier pour l’obtention de son financement

Je remercie également mes camarades de promo, Yann, Arlinda, Sophie et Léa pour l’entraide bienveillante dont ils ont tous fait preuve et ma femme Mireille, qui a su me soutenir et m’encourager dans les moments de doutes et de remises en question.

**Liste des compétences couvertes par le référentiel**

1. Maquetter une application
2. Réaliser une interface utilisateur web statique et adaptable
3. Développer une interface utilisateur web dynamique
4. Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce
5. Créer une base de données
6. Développer les composants d’accès aux données
7. Développer la partie back-end d’une application web ou web mobile
8. Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

Résumé de projet

L’idée de ce projet est née après une discussion avec ma fille étudiante en master 1 en environnement à l’université Paul Valéry de Montpellier 3 qui m’évoque une association d’étudiants au sein de sa faculté

Cette association a pour but de présenter des sujets environnementaux, écologiques auprès de la communauté universitaire mais aussi de présenter le Master, sur les 2 années, Gestion de L’Environnement (GE), ainsi que les droits des étudiants, ….

Elle expose notamment l’idée du jardin partagé mis en place par l’association mais qui a du mal à se développer de façon pérenne, en évoquant peut être le fait que l’association n’ait pas encore acquis suffisamment de visibilité et de notoriété. D’où l’idée sous-jacente du bureau de l’association de pouvoir développer la communication à travers un site d’information sur le master Gestion de l’environnement et sur les différents projets écologiques menés par l’association et les étudiants de Master dans le cadre de leur cursus.

Sensible aux problématiques environnementales, et soucieux de me rendre utile à un groupe de jeunes étudiants, l’idée m’a vite séduit et j’ai donc proposé la création d’un site au bureau administratif pour mettre en valeur leur association et leurs projets.

Cette démarche s’inscrit aussi dans l’idée de moderniser l’image de l’association , d’attirer de nouveaux adhérents, d’augmenter la visibilité du Master et de pérenniser les supports dans le temps.

Afin de pouvoir échanger sur les différents aspects du projet et sur sa faisabilité, nous avons pratiqué des échanges via Zoom.

Suite à l’analyse de leurs besoins définis en équipe de leur côté, j’ai pu leur proposer un **cahier des charges**, en exposant les contraintes techniques et les limites du projet, puis par la réalisation de la maquette graphique (des wireframes à la maquette HTML/CSS) et enfin tout le développement du site avec le déploiement des composants d’accès à la base de données et l’ajout au fur et à mesure des différentes fonctionnalités qui permettent aujourd’hui aux clients étudiants d’être quasiment 100% autonomes dans la gestion de leur site : ajout, modification et

suppression d’actualités, d’offres d’emploi, d’outils numériques et la gestion des utilisateurs par les administrateurs étudiants désignés.

Au niveau de l’environnement et des technologies utilisés, j'ai développé mon projet avec l'éditeur de code « Visual studio code ». Le langage de programmation est le PHP et JavaScript sous le framework Symfony 5.4, avec le support du framework Bootstrap pour le front-end. Je me sers également de l'ORM Doctrine, inclus dans Symfony pour la création et la gestion de la base de données. La plateforme de développement est WampServeur avec phpMyAdmin pour l'administration Web des bases MySQL.

Ce projet fut enrichissant, car il m’a permis de m’adapter en fonction des besoins évolutifs de mes interlocuteurs.

Le projet et son déroulement furent très appréciés par le bureau de l’association des étudiants sachant que ce site pour répondre complètement aux besoins de l’association qui ne cessent d’évoluer, devra être développé davantage, ce que je pourrai prétendre une fois ma formation finie afin de leur proposer un produit plus complet.

Environnement de travail

**Les outils utilisés**

Au cours de son développement, le projet a nécessité l’utilisation de nombreux outils.

Maquettage et diagrammes

|  |  |
| --- | --- |
|  | L’application google **draw.io**, pour construire les diagrammes d’activité, UML, ainsi que les wireframes du site |
|  | **Looping**, un logiciel gratuit et libre d’utilisation, qui a été conçu par l’Université Toulouse III, et qui permet notamment de construire des MCD (Modèles Conceptuels de Données) et des MLD (Modèles Logiques de Données), conçus à partir de la méthode Merise |

Back-end

|  | **Symfony** est un framework [PHP](https://mobiskill.fr/emploi/developpeur-php/) open source avec une architecture MVC. C’est l’un des framework les plus populaires parmi la communauté des développeurs open source. Il est utilisé pour construire des applications web complexes et performantes Un framework a pour but de simplifier et d’accélérer le processus de développement. Ce framework PHP présente l’avantage d’être puissant et robuste. |
| --- | --- |
| logo-twig.png | **Twig** est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le framework Symfony. |
|  | **PHP** est le langage de programmation associé à Symfony |
|  | **Composer** me permet de déclarer et d'installer les bibliothèques dont le projet a besoin. |
|  | La plate-forme de développement web **Wampserver**, permet de simuler localement le fonctionnement du site internet. |
|  | En ce qui concerne la base de données j’ai utilisé **MySQL** ( My Structured Query Language ) qui est un gestionnaire Open-Source couplé à PHPMyAdmin une interface simple et facile d’accès. |
|  | Pour la mise en ligne du site, j’ai utilisé **WinSCP**, un client SFTP (SSH File Transfer Protocol). |
| cpanel.png | **cPanel,** un panneau de configuration permettant de gérer l’hébergement d’un site web. |

**Front-End**

| twbs/bootstrap-icons - Packagist | J’ai utilisé le framework HTML **Bootstrap** lors de la conception de la maquette du site en HTML/CSS.  Bootstrap est une collection d’outils utiles à la création du design de sites et d’applications web. C’est un ensemble qui contient des codes HTML et CSS, des **formulaires, boutons, outils de navigation et autres éléments interactifs et également** une bibliothèque d’icône, ainsi que des extensions JavaScript en option. Son système de Grid est particulièrement efficace. |
| --- | --- |
|  | Le préprocesseur CSS (Cascading Style Sheet) **Sass**, m’a permis d’organiser et de construire mon CSS plus aisément, de faciliter les mises à jour en cas de modification du code |
|  | Pour les effets dynamiques du site j’ai utilisé le langage JavaScript et sa bibliothèque « jQuery ». |
|  | **Ajax** pour Asynchronous JavaScript and XML, correspond à un groupe de méthodes et de moyens visant à permettre d'établir une communication asynchrone entre le navigateur et le serveur. **Ajax** permet d'effectuer des modifications parcellaires sur une page web, sans recharger l'ensemble de la page internet. |

**Logiciels**

|  |  |
| --- | --- |
|  | Concernant mon environnement j’ai développé sur l’IDE (Integrated Development Environment ou Environnement de Développement) **Visual Studio Code**, qui permet de personnaliser facilement son espace de travail, et les nombreuses extensions que l’on peut lui adjoindre de rendre plus facile le travail de codage.  Coté terminal j’ai utilisé powershell qui est intégré à Visual Studio Code, couplé à la gestion de dépendance composer il m’a permis la construction de mon projet, le téléchargement des différents bundles ainsi que d’effectuer les mises à jour nécessaire |
|  | J’ai utilisé la plateforme d’hébergement **Github**, afin de faire des sauvegardes en cas de mauvaise manipulation ou de problèmes techniques et de son interface Github Desktop |
|  | Afin de créer certains designs, de redimensionner certains contenus multimédias, ou de produire les différentes icônes, j’ai utilisé le logiciel libre et gratuit de retouche d’images **Gimp** |

Cahier des charges

**Analyse de l'existant**

Il s’agit d’un nouveau projet de création de site pour l’association Paul Green. Un site existait auparavant mais était devenu obsolète, sans aucune maintenance, ni suivi et les codes administrateurs étaient inconnus depuis plusieurs années par les étudiants.

### Définition de projet et Cahier des charges

Après plusieurs échanges en visioconférence et par mail, les clients ont mis en évidence les éléments importants que le projet final devait respecter (annexe 1). Étant donné l’ampleur des besoins exprimés, j’ai pu donc préparer un cahier des charges non exhaustif où j’ai repris les principaux points dans l’objectif de l’examen, sachant que le site fera l’objet d’une continuité après la formation pour répondre au mieux aux besoins des étudiants.

**Les objectifs du site que je me suis fixés pour l’examen**

Rendre les informations de l’association et du master GE accessibles et permettre la compréhension des actions menées et les projets collectifs du Master 1 et 2 par le plus grand nombre, que ce soit les étudiants du Master ou tout autre étudiant

Permettre aux adhérents d’accéder à un espace adhérents pour y trouver des informations précises tout en limitant l’accès selon leurs statuts (membres du bureau, adhérents simples membres du Master GE et pour les autres étudiants du campus non membres du Master GE. )

C'est un site à vocation informative (palier au manque de visibilité de l’association ) mais aussi contributif (apporter des idées, prendre contact, consulter des documents)

Je souhaite mettre en avant l’association, la présentation du master GE et les actions menées par l’association. Puis y ajouter une rubrique contact pour joindre l’association et un espace sur les partenaires de l’association.

Enfin, un espace de promotion des actions locales pour l’environnement sur le bassin montpelliérain.

Les visiteurs visés sont principalement des étudiants de l’université Paul Valéry Montpellier 3 en master GE (actions, offres de stage, anciens travaux) mais aussi à toute autre étudiant sensible aux problématiques de l’environnement et de l’écologie mais aussi de façon plus large aux futurs étudiants intéressés par le Master GE.

**Périmètre du projet**

Mon site ne doit pas être multilingue.

Le site internet doit être **évolutif**, et pouvoir intégrer les différents **réseaux sociaux** et médias au fur et à mesure que l’association développera sa communication.

Le contenu doit pouvoir être modifié et ajouté par les **membres du bureau de l’association**, et la partie back-end se devait donc d’être **accessible** et facilitée pour des néophytes en développement et gestion de sites web.

D’un point de vue général, le site devra être **moderne**, pour atteindre une cible de jeunes étudiants mais aussi pour montrer **la dynamique** du secteur de l’environnement.

**Graphisme et ergonomie**

Aucune charte graphique, tout était à définir !

Le **logo** existant allait servir de guide pour concevoir le design, qui se devait d’être **accueillant et épuré**, et rappeler les **activités du site.**

Veille Concurrentielle

Afin d’établir une charte graphique du site web et de correspondre au mieux à mes attentes et à celles de mes clients, j’ai effectué différentes recherches afin de trouver des renseignements sur divers sites que ce soit associatif, environnemental, ou universitaire. Mes recherches m’ont beaucoup aidé à trouver des solutions design et efficaces pour répondre aux besoins de l’association, mais aussi à approfondir le contenu pour être au plus près des requêtes des membres du bureau de l’association.

Voici les différents sites qui m’ont beaucoup aidé pour le design et les fonctionnalités que les sites ont tendances à utiliser :

**Des associations d’étudiants sur l’environnement :**

<https://www.helloasso.com/associations/association-des-etudiants-en-droit-de-l-environnement-aede/evenements/tote-bags-100-recycles-aede>

<https://artsetmetiers.fr/index.php/fr/actualites/environnement-quand-les-etudiants-sengagent>

<https://www.univ-nantes.fr/sepanouir-sur-les-campus/vie-associative-et-engagement/les-associations-etudiantes-environnement>

<https://le-reses.org/>

**Des sites de présentation du master GE**

<https://master-act-alumni.jimdofree.com/>

<https://esigat.wordpress.com/stages/>

<https://www.etudiants-mediation-scientifique.com/association-s-e-n-s-symbiose-entre-nature-et-societe/>

<https://anestaps.org/nos-partenaires/>

<https://www.master-gedd-unice.fr/Association-GAIA.htm>

**Des sites d’associations environnementales**

<http://www.natur-action.org/partenaires/>

<https://fne.asso.fr/>

**Autres sites :**

<https://www.cite-sciences.fr/fr/au-programme/lieux-ressources/bibliotheque/quoi-faire/evenements/environnement-les-initiatives-etudiantes/>

Charte graphique

**Pour le choix de la typographie :**

L’étude du logotype m’a servi de référence pour choisir la typographie du site pour le texte et pour les titres en me rapprochant des polices utilisées par les étudiants à l’origine de la refonte du logo.

**Pour le texte **



**Trebuchet**

**Pour les titres :**

**Pour les titres :**

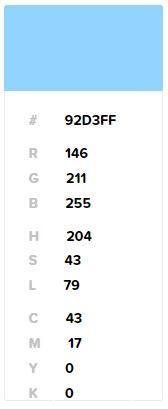
**Gelasio**

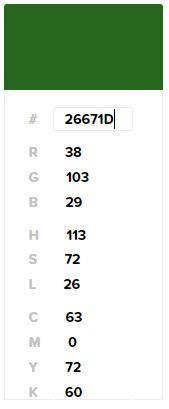
**Parisienne**

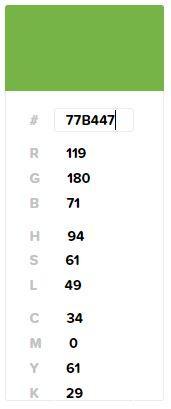
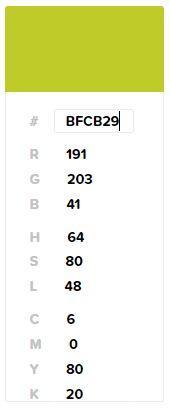


**Pour le choix des couleurs :**

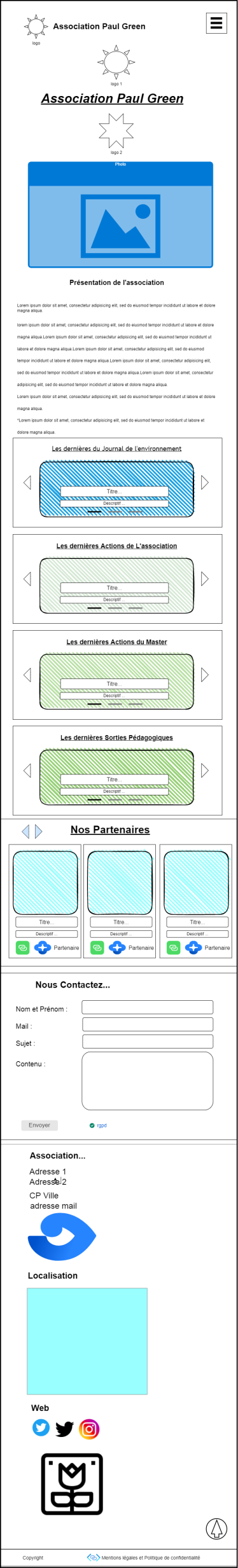
Le chemin directeur pour le choix des couleurs est également parti du découpage du logo fourni par l’association.

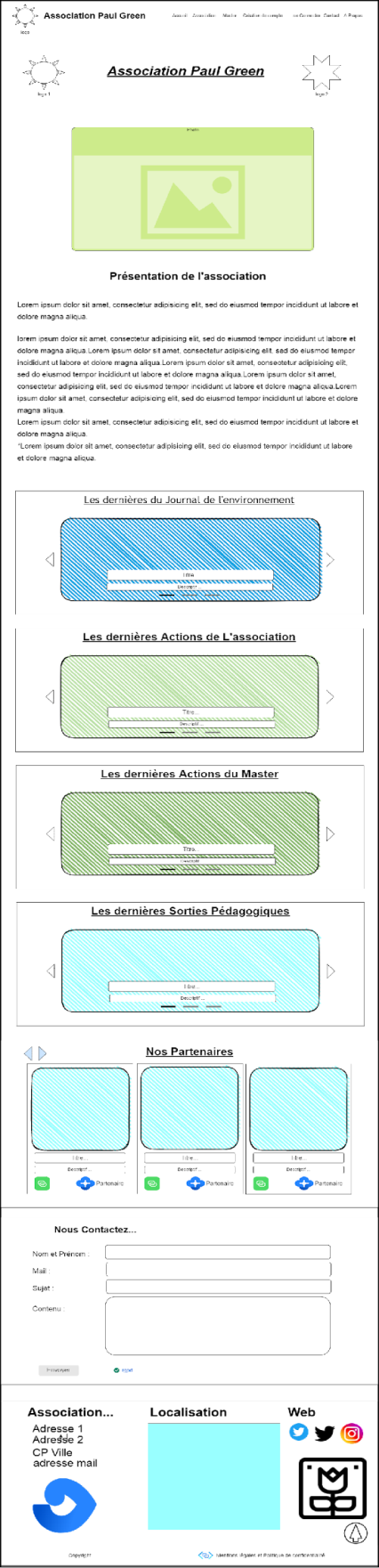
Un dégradé de couleur de couleur peut être effectué en partant de ces couleurs principales rappelant les thématiques de l’environnement (nature, eau)





Wireframe et Maquettage

**Maquetter une application**

J’ai utilisé l’application « diagrams.net ». Elle est une solution en ligne, gratuite, qui permet la création de schémas, de diagrammes, d’organigrammes. Ici, elle me sert à la génération des premiers wireframes, ceux de la page d’accueil, aussi bien en mode PC qu’en mode mobile ou responsive.

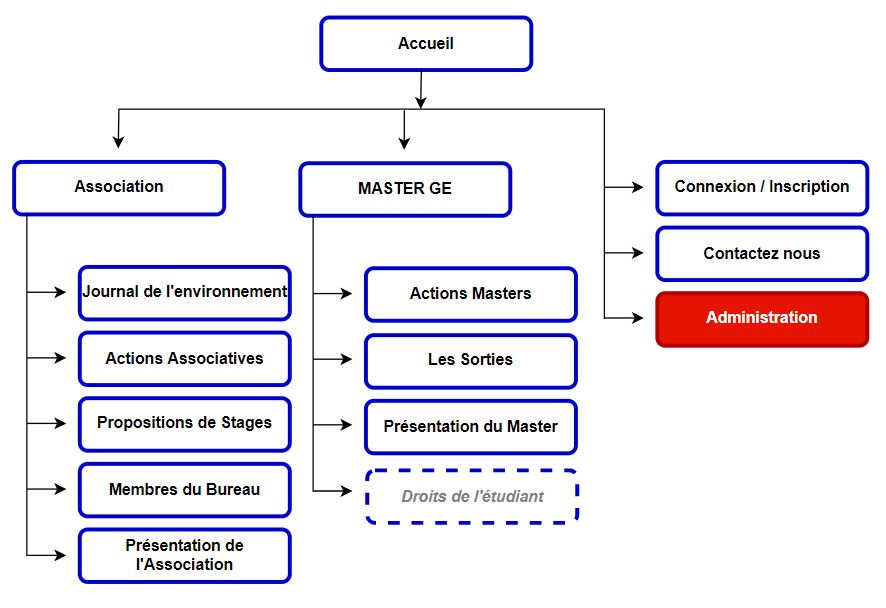
À gauche nous avons la wireframe de la page d’accueil.

À droite nous avons la wireframe de la page d’accueil responsive.

.

Arborescence du site

**Voici l’arborescence du site:**



**Arborescence détaillée**

Chaque page du site contient plusieurs éléments dont voici les principaux :

**La page Accueil :**

- Présentation de l’association

- Par un système de carrousel,

* présentation des 5 dernières nouvelles du journal de l’environnement
* présentation des 5 dernières actions menées par l’association
* présentation des 5 dernières actions menées par le Master
* présentation des 5 dernières sorties pédagogiques organisées par le master
* présentation des partenaires de l’association

- Avoir la possibilité d’avoir un système de contact

**L’ensemble des sections** se présente sensiblement de la même manière pour les utilisateurs

* Un titre
* Une photo principale
* Différentes informations, dates,…
* D’autres photos non mises en avant,
* Des liens sur des sites ou des documents…

**L’affichage du bureau** avec photos et informations consultables.

**La partie connexion** des utilisateurs et la **déconnexion.**

**La partie administration :**

Cette partie est réservée aux seules personnes connectées et définies en tant qu' administrateur

Elle permet la modification de pratiquement l’ensemble des données affichées sur le site, ainsi que la gestion complète des différents items.

* La gestion complète des utilisateurs et de leurs rôles pouvant avoir lieu au sein de l’association.
* La présentation du Master, ou de l’association se fait par l’intermédiaire d’une zone de texte entièrement libre réalisée avec le bundle CKEditor.
* Les actions du Master et de l’Association, les sorties, les offres de stages ou de travail utilisent pour les textes longs ce même bundle. D’autres informations prédéfinies sont intégrées. Des photos, des liens sur des documents ou sites viennent compléter les différentes sections. Une gestion complète de ces items est proposée.

Réalisation du site

**Planification du projet**

J’ai réalisé une liste de tâches détaillées. Je tiens tout de même à notifier qu’il m’aurait été possible de réaliser un diagramme de Gantt, mais n’étant pas encore familiarisé avec le Framework Symfony j’étais dans l’incapacité de quantifier correctement mon temps utilisé pour les différentes parties du site.

**Back-End :**

* Définir les rôles utilisateurs.
* Création de diagrammes.
* Création du squelette Symfony.
* Création de la base de données.
* Création des entités avec leurs relations.
* Mise en place du service global d’authentification utilisateur.
* Sécurité du site et des utilisateurs.
* Mise en place de la gestion de cookies, reCAPTCHA.
* Test de sécurité

**Front-End :**

* Mise en place du framework CSS Bootstrap.
* Travail sur la partie Header.
* Travail sur la partie Footer.
* Intégration des différents blocs en fonction des pages.
* Travail sur le responsive design.

**Finalisation :**

* Passage en production.
* Phase de test.
* Correction des bugs.
* Test du site en version mobile.
* Livraison du site.

**Définition des rôles**

Pour ce qui concerne les rôles des utilisateurs ayant un compte, quatre niveaux sont prévus :

L’administrateur (ADMIN), les comptes Association Paul Green, les comptes Master GE et les utilisateurs Autres.

**L’Administrateur**

Les administrateurs sont les seuls qui peuvent modifier l’ensemble des données, modifier les présentations de l’association, du master, de gérer complètement les utilisateurs, les différentes fiches actions associatives, du master, des sorties, des stages…

L’ensemble des comptes suivants ne peuvent que consulter les informations saisies par les administrateurs.

**Les comptes de l’Association Paul Green**

Ces comptes ont accès à l’ensemble des informations.

**Les comptes du Master GE**

Ces comptes n’ont accès qu'à la visualisation des informations concernant directement le master, c'est-à-dire, les actions du Masters, les sorties pédagogiques, la présentation du master.

**Le compte Autres**

Ces comptes n'accèdent qu'aux informations concernant l’association, c’est-à-dire la présentation de l’association, les actions de l’association, les proposition de stages

**Les visiteurs**

Les visiteurs, sans compte, n'accèdent qu’à la page accueil.

MCD / MLD / UML

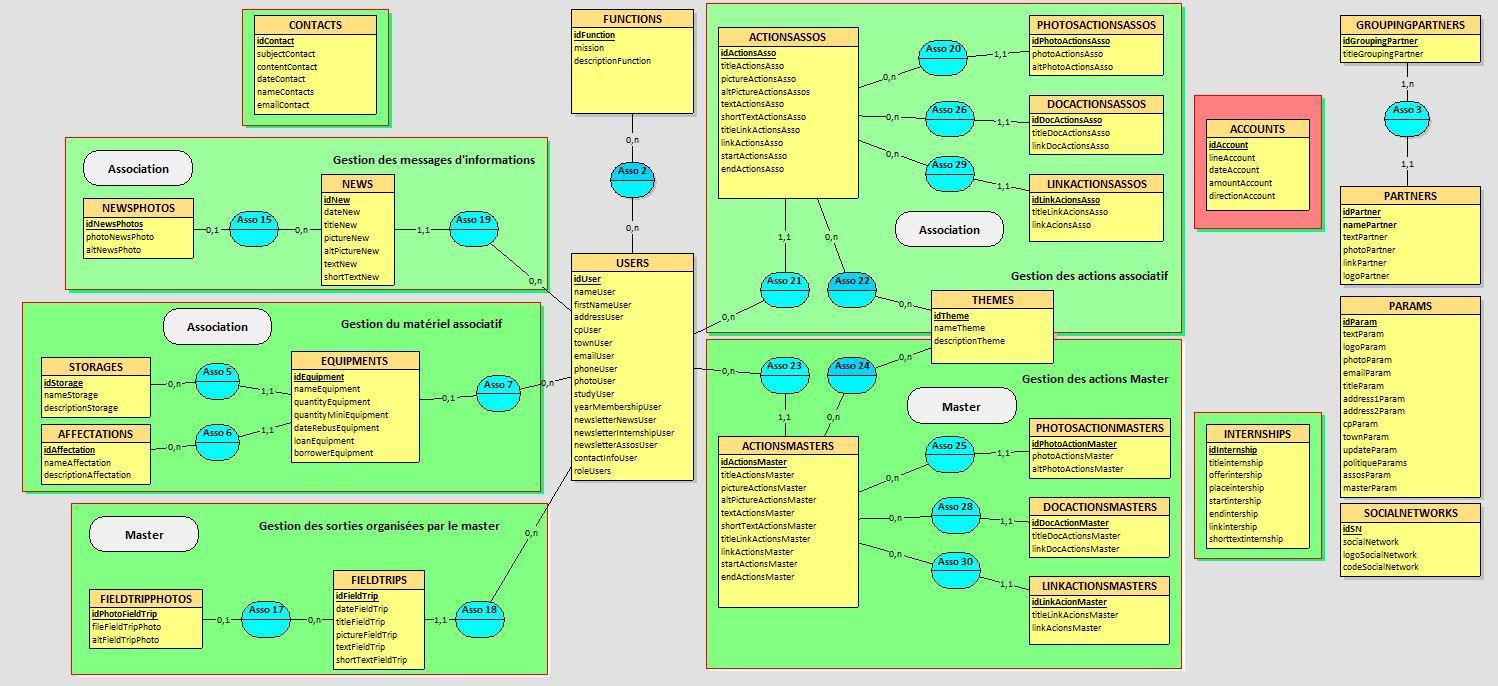


Afin de construire les **MCD (Modèles Conceptuels de Données)** et les **MLD (Modèles Logiques de Données**, j’ai utilisé le logiciel **Looping**. Ce logiciel est gratuit et libre d’utilisation. Il permet la modélisation des entités avec leurs propriétés et types de données, les associations et cardinalités des liens entre les entités. Il est une aide importante pour la réalisation de la base de données.

**MCD**

Suite à l’étude des demandes pour la réalisation du site, et des différentes informations à stocker, il est possible de réaliser le MCD ( Modèle Conceptuel de Données ). Le logiciel Looping permet de préparer la conception des entités, des propriétés, des caractéristiques des propriétés, et des liens entre entités. La structure globale de la base de données d’une manière schématique est ainsi prête pour un transfert vers la base de données réelle.

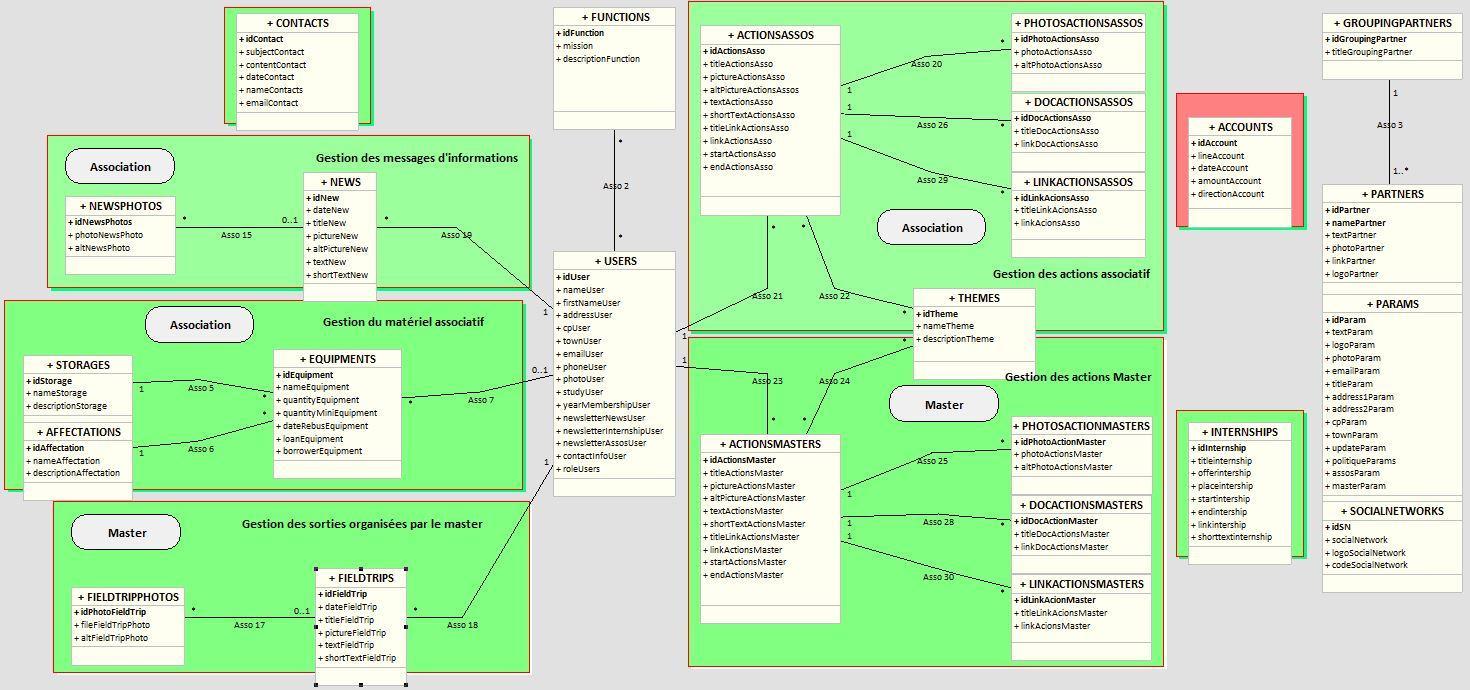
Les entités sont en Majuscule et les propriétés sont en camel case, 1er mot tout en minuscule et les suivants attachés, en minuscule avec l'initiale en majuscule.



**UML**



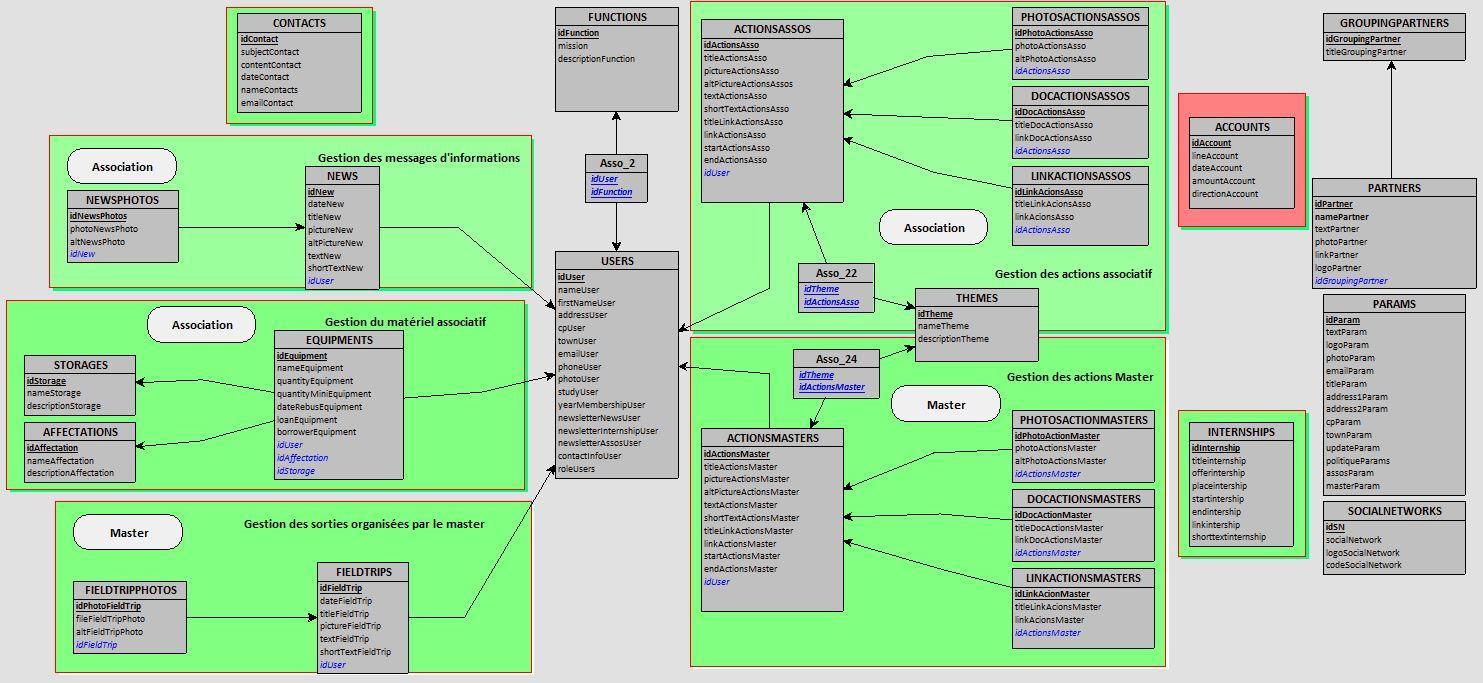
L’ UML ( Langage de Modélisation Unifié ) est généré automatiquement à partir du MCD réalisé.



**MLD**

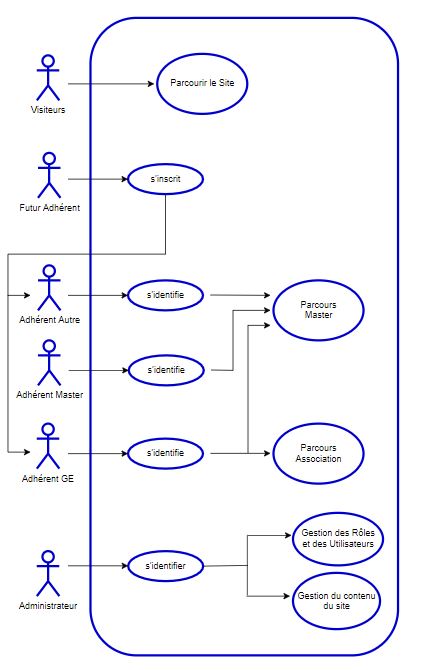


De même, le MLD ( Modèle Logique de Données ) est réalisé automatiquement. Ce dernier permet de voir la création des tables intermédiaires (tables pivots), entre deux entités ayant des cardinalités 0-1,n / 0-1,n.



**Use Case Diagramme**

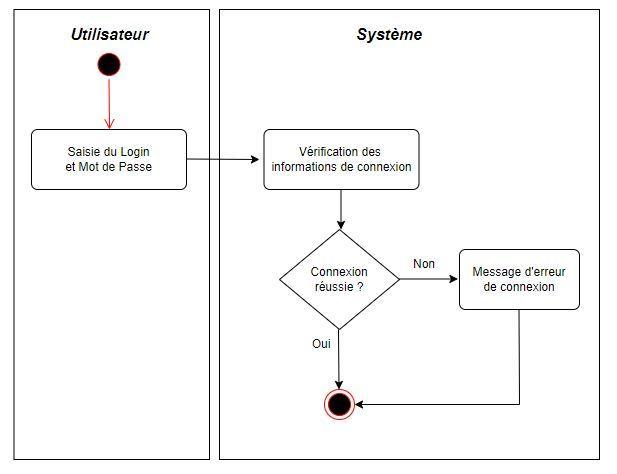
Afin d’avoir une vision globale du comportement fonctionnel du site en fonction de chaque rôle utilisateur, je réalise un diagramme de cas d’utilisation.



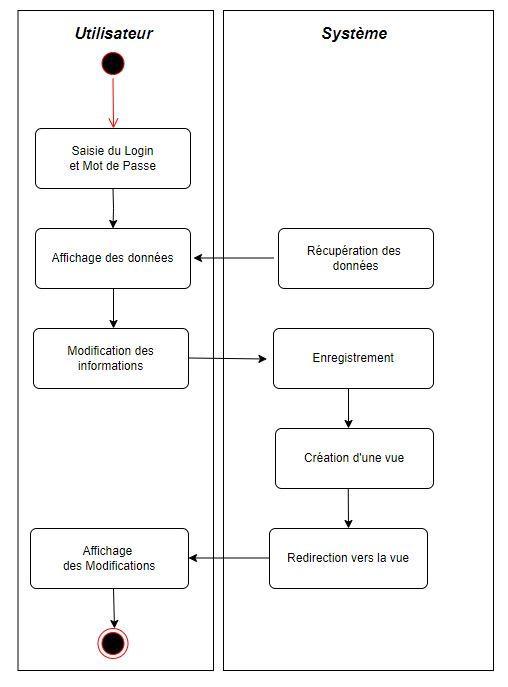
**Diagrammes d’activités**

Afin de pouvoir visualiser de manière beaucoup plus précise les interactions entre usager et l’interface du site j’ai conçu des diagrammes de séquences. Voici deux de mes diagrammes d’activités :

**Diagramme D’activite Connexion Utilisateur**



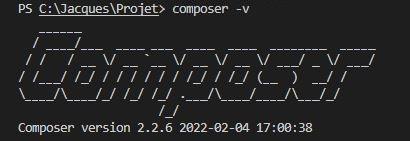
**Diagramme D’activite Modification De Contenu**



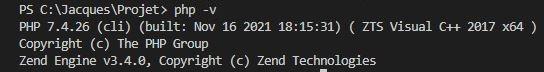
Développement Back-End

**Création de la base du squelette et de la base du projet**

Pour commencer correctement le projet avec le framework symfony, il faut vérifier en premier lieu que les logiciel PHP et Composer soit correctement installés et utilisable sur Visual Studio Code, l’IDE utilisé.

Le logiciel Composer est un gestionnaire de dépendances et permet à ses utilisateurs de déclarer et d’installer les bibliothèques requises pour notre projet.

Le PHP (PHP : Hypertext Preprocesseur) est notre langage de programmation principal, associé au Twig de Symfony et d’autres…

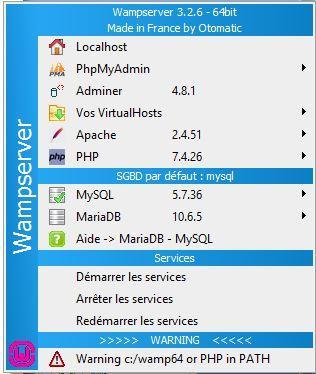


La ligne de commande suivante, grâce au logiciel Composer, installe Symfony, notre socle pour notre projet. Il crée le répertoire « AssociationPaulGreen », avec l’ensemble des modules pour Symfony.

PS C:\Jacques\Projet> composer create-project symfony/website-skeleton AssociationPaulGreen

Afin de démarrer le serveur web de développement, du serveur interne de php, il nous suffit de lancer la commande ci-dessous :

PS C:\Jacques\Projet\AssociationPaulGreen> php –S 127.0.0.1.8000 –t public

Pour le fonctionnement, il a bien sûr fallu installer au préalable un serveur web (type Apache, Nginx). Pour notre part nous avons installer la plateforme de développement web de type Wamp, « WampServer ».C’est un environnement comprenant trois serveurs (Apache, MySQL et [MariaDB](https://fr.wikipedia.org/wiki/MariaDB)), un interpréteur de script (PHP), ainsi que [phpMyAdmin](https://fr.wikipedia.org/wiki/PhpMyAdmin) pour l'administration Web des bases MySQL. comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL.

Symfony étant en POO (Programmation Orientée Objet), il faut créer un ‘controller’ qui va permettre de retourner une réponse à une route ou une requête donnée. On rentre alors dans le terminal la commande suivante pour créer notre controleur “MainController” :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:MainController

**Construction de la base de données**

Symfony utilise par défaut l’ORM Doctrine.

Afin que le projet soit opérationnel, il faut créer une base de données de type MySQL.

Il nous faut modifier le fichier **.env**, à la racine du projet.

La variable d’environnement **« DATABASE\_URL »** est modifiée afin d’obtenir une relation entre mon projet et la base de données sous MySQL.

###> doctrine/doctrine-bundle ###

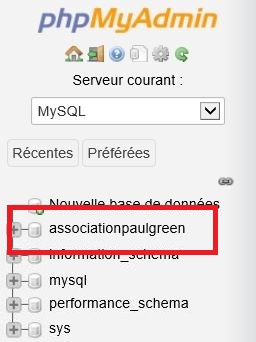
DATABASE\_URL=mysql://root:@127.0.0.1:3306/AssociationPaulGreen?serverVersion=5.7&charset=utf8mb4

###< doctrine/doctrine-bundle ###

Le terme dans la commande Database, « AssociationPaulGreen » est alors le nom de base de données dans MySQL.

Pour la créer, il faut lancer la commande suivante

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console doctrine:database:create



Il est important de vérifier si elle a bien été créée dans PhpMyAdmin.

**Création des entités**

**Création de l’entité Users**

Sur Symfony, la construction des entités se fait par le biais de ligne de commande dans le terminal à la racine du projet et bénéficie d’une trame chronologique nous demandant sa structure avec les paramètres des champs et les relations entre tables.

La première entité à créer est l’entité pour les utilisateurs du site : Users. Elle permet la gestion complète des personnes connectées, et de définir les rôles et droits de chacun.

Elle est spécifique à la sécurité. Elle encode les mots de passe, et gère l'authentification pour la connexion au site, et l’enregistrement des utilisateurs.

Pour générer l’entité User, et toute la gestion de la sécurité, la ligne de commande suivant est lancée :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:user

Pour nous ici, le nom de l’entité est Users.

Cette commande nous crée l’ensemble des modules dans notre répertoire de notre projet pour la gestion de la sécurité.

Pour la table dans la base de données, il suffit de migrer l’entité vers la base de données.

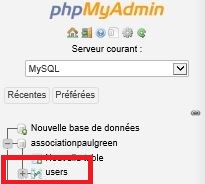
Les lignes de commande dans le terminal suivantes lancent la mise à jour de la base données.

Le fichier de migration de la classe, créé par la commande suivante, liste une suite de requêtes SQL afin de modifier la base de données en fonction de l’entité préalablement préparée dans le projet, et suivant l’existant.

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:migration

Puis, cette commande exécute ces requêtes SQL du fichier de migration.

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console doctrine:migrations:migrate



En vérifiant la base de données, nous constatons que la table users à bien été générée dans la base de données.

**Construction de l’authentification**

Grâce aux lignes suivantes saisies dans le terminal, Symfony créer un système authentification par email et mot de passe.

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:auth

Une fois rentré les bons paramètres, il suffit de mettre à jour la base de données de la même manière que précédemment :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:migration

Puis :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console doctrine:migrations:migrate

J’ai par la suite rajouté une double vérification de mot de passe dans le formulaire d’enregistrement sur le site.

j’ai utilisé aussi deux bundles afin d’apporter un plus d’ergonomie :

Pour les installer j’ai utilisé les lignes de commande suivantes :

PS C:\Jacques\Projet\AssociationPaulGreen> composer require symfonycasts/reset-password-bundle

Et

PS C:\Jacques\Projet\AssociationPaulGreen> composer require symfonycasts/verify-email-bundle

Le premier Bundle « reset password » créer un système de reset de mot de passe en cas de perte ou d’oubli.

Le second permet de rajouter une vérification par email de ce changement de mot de passe.

**Construction Du Formulaire D’enregistrement**

La création de ce formulaire de saisie se fait d’une manière automatique par l’intermédiaire de la commande du terminal suivante :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:registration-form

Un formulaire d’inscription est ainsi initié.

Il est bien entendu qu’il est maintenant important d’adapter l’ensemble des entités, classes, de la base de données, des tables aux besoins du projet. Sur la classe Users, des champs supplémentaires sont à prévoir.

De même les redirections de page sont à modifier suivant les besoins.

Des droits d’utilisation du site sont élaborés suivant ce qui a été défini par les étudiants de l’association. Ces droits sont ajoutés à l’entité Users.

use Symfony\Component\Security\Core\User\UserInterface;

use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;

/\*\*

 \* @ORM\Entity(repositoryClass=UsersRepository::class)

 \* @UniqueEntity(fields={"email"}, message="Il existe déjà un compte avec cet email !")

 \*/

class Users implements UserInterface, PasswordAuthenticatedUserInterface

{

    public const ROLE\_USER = 'ROLE\_USER';

    public const ROLE\_OTHER = 'ROLE\_OTHER';

    public const ROLE\_GE = 'ROLE\_GE';

    public const ROLE\_ADMIN = 'ROLE\_ADMIN';

    public function \_\_construct()

        {

        $this->roles = [self::ROLE\_USER];

        }

Ces différents niveaux de droits sont contrôlés par l’annotation @isGranted ou la fonction is\_Granted().

Par exemple, la partie des équipements est réservée exclusivement aux Administrateurs :

<?php

namespace App\Controller;

use App\Entity\Equipments;

use App\Form\EquipmentsType;

use App\Repository\EquipmentsRepository;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Annotation\Route;

/\*\*

 \* @Route("/equipments")

 \* @isGranted("ROLE\_ADMIN")

 \*/

class EquipmentsController extends AbstractController

{

    /\*\*

     \* @Route("/", name="app\_equipments\_index", methods={"GET"})

     \*/

    public function index(EquipmentsRepository $equipmentsRepository): Response

    {

Dans le cas ou une personne saisissait l’adresse  « /equipments » directement sans avoir les droits admin, elle serait renvoyée vers la page de connexion.

Ou par exemple dans le fichier Twig du menu :

                    {% if is\_granted(«ROLE\_ADMIN») %}

                        <li class="nav-item dropdown">

                            <a class="nav-link dropdown-toggle"

                                    href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

**Création Des Autres Entités**

L’ensemble des entités est créé sur le même principe.

La ligne de commande lance la création des entités dans notre projet.

Pour créer une entité avec Symfony, il suffit d’écrire dans le terminal la commande :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:entity

Un questionnaire nous permet alors de définir ses propriétés, et les caractéristiques de celles-ci.

Exemple détaillé pour l’entité SocialNetworks :

Class name of the entity to create or update (e.g. BraveElephant):

> SocialNetWorks

created: src/Entity/SocialNetWorks

created: src/Repository/SocialNetWorksRepository.php

Entity generated! Now let's add some fields!

You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):

> socialNetWork

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 50

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/SocialNetWorks.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> logoSocialNetWork

Field type (enter ? to see all types) [string]:

>

Field length [255]:

>

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/SocialNetWorks.php

> codeSocialNetWork

Field type (enter ? to see all types) [string]:

>

Field length [255]:

>

Can this field be null in the database (nullable) (yes/no) [no]:

>

updated: src/Entity/SocialNetWorks.php

> linkSocialNetWork

Field type (enter ? to see all types) [string]:

>

Field length [255]:

>

Can this field be null in the database (nullable) (yes/no) [no]:

>

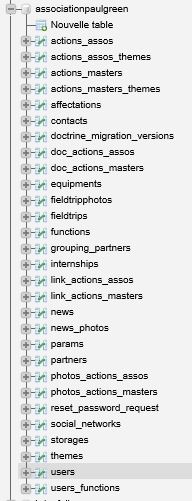
updated: src/Entity/SocialNetWorks.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

Success!

Next: When you're ready, create a migration with make:migration



Après avoir rentré les bons paramètres, il suffit de mettre à jour la base de données de la même manière que précédemment :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:migration

Puis :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console doctrine:migrations:migrate

La base de données est ainsi mise à jour.

Pour chaque entité, nous définissons ses propriétés, puis nous synchronisons avec la base de données.

Pour être clair, les différentes ‘**entités**’, sont autant de ‘**classes**’ dans Symfony, et les **propriétés** qui les caractérisent autant de ‘**fonctions**’ dans ces ‘classes’.

***Manipulation des données – La fonction CRUD***

Symfony nous permet de générer une suite de fichiers pour la gestion complète des entités. La fonction CRUD (Created, Read, Update, Delete) nous construit ces fichiers et nous pouvons l’appliquer sur l’ensemble de nos entités modifiable par les utilisateurs. Elle crée l’ensemble des fonctions de gestion dans les « controllers » ainsi que les formulaires liés à nos entités, et les « templates » associées .

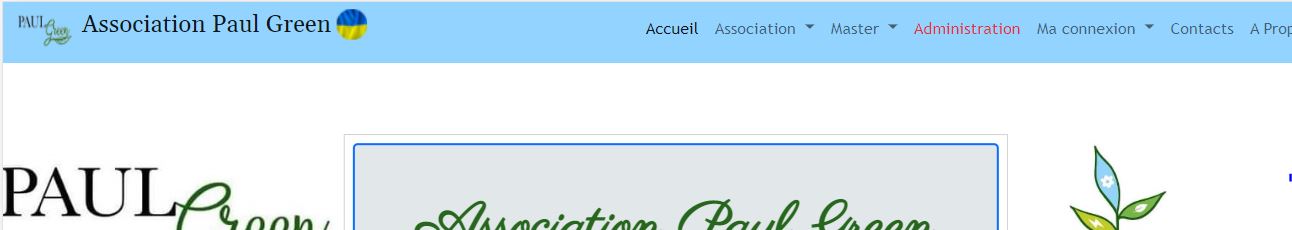
La commande suivante est lancée pour chaque entité que nous voulons gérer :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/console make:crud

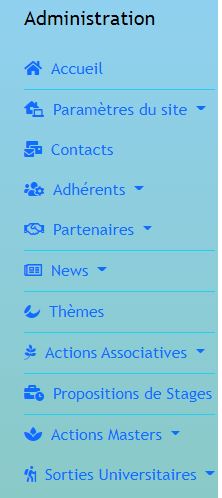
Certaines entités ne sont pas gérer par ce biais.

Une fois ces CRUD réalisé, nous pouvons adapter plus précisément les gestion, améliorer les qualités de présentation des informations.

Les administrateurs sont les seuls users à pouvoir modifier les données. Une fois connecté, ils ont accès à un nouveau menu Administration, leur permettant la gestion complète.



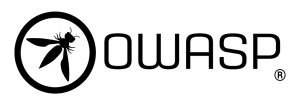
Puis :

En cliquant sur Administration(en rouge) sur la barre de menu principal,

Les utilisateurs ont accès à la gestion complète des données. Les gestions des entités liés sont souvent regroupée.

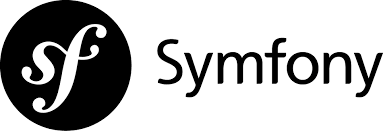
##### **Sécurisation et tests**

##### **Veille sur la sécurité**

Pour effectuer ma veille de sécurité, j’ai consulté le site de l’[OWASP](https://owasp.org/) (Open Web Application Security Project), une communauté en ligne travaillant sur la sécurité des applications Web. Il publie régulièrement des recommandations de sécurité et explique aux internautes comment sécuriser leurs projets. Il présente les dix principales attaques sur les sites.

<https://owasp.org/www-project-top-ten/>

Dans la documentation Symfony, une partie est réservée à la sécurité.

<https://symfony.com/doc/current/security.html>

J’ai consulté aussi certains blogs dont celui de Wanadev, et en autre pour article « 5 sources d'amélioration pour la sécurité de votre application Symfony ».



<https://www.wanadev.fr/78-5-sources-d-amelioration-pour-la-securite-de-votre-application-symfony/>

Ou encore le blog de Vaadata, une société spécialisée dans le ‘pentest’, les tests d’intrusion.

<https://www.vaadata.com/blog/fr/>

##### **Tests Unitaires de Sécurité**

Pour effectuer des tests unitaires j’ai installé le Framework Open Source PHPUnit.

Ce dernier est dédié au langage de programmation PHP et nous permet de tester certaine fonctionnalité du site internet.

Pour cela on utilise Composer et on rentre les lignes de commande suivantes :

PS C:\Jacques\Projet\AssociationPaulGreen> composer require --dev symfony/phpunit-bridge

Puis :

PS C:\Jacques\Projet\AssociationPaulGreen> composer require --dev symfony/browser-kit symfony/css-selector

Pour tester les “controllers” et vérifier que toutes les routes que l’on a définies mènent bien à une page, je crée un dossier ‘Controllers’ dans le dossier ‘test’ (présent dans l’architecture de base d’un projet Symfony), dans lequel j’ajoute un nouveau fichier pour chacune de mes entités.

Exemple pour l’entité ‘Themes’ :

<?php

//tests/Controller/ThemesControllerTest.php

namespace App\Tests\Controller;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class ThemesControllerTest extends WebTestCase

{

public function testNewThemes()

{

$client=static::createClient();

$client->request('GET','/themes\_new');

$this->assertEquals(200, $client->getResponse()->getStatusCode());

}

}

Ici nous avons un test pour vérifier si la page d’ajout d’un nouveau thème existe et fonctionne correctement.

Pour mes Formulaires :

Je cherche également à tester mes formulaires. Pour cela, dans la même logique que pour les “controllers”, je crée dans ‘tests’ un dossier ‘Form’, et j’ajoute un nouveau fichier pour chaque formulaire que je souhaite tester. Par exemple pour l’entité Internships’ (mon formulaire des stages) :

<?php

namespace App\Test\Form;

use Datetime;

use App\Entity\Internships;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class InternshipsFormTest extends KernelTestCase{

public function testInternships(){

$date=new Datetime();

$interships=(new Internships())

->setOfferinternship('77')

->setPlaceintership('77')

->setStartinternship($date)

->setEndinternship($date)

->setLinkinternship('77');

self::bootKernel();

$error = self::$container->get('validator')->validate($interships);

$this->assertCount(0,$error);

}

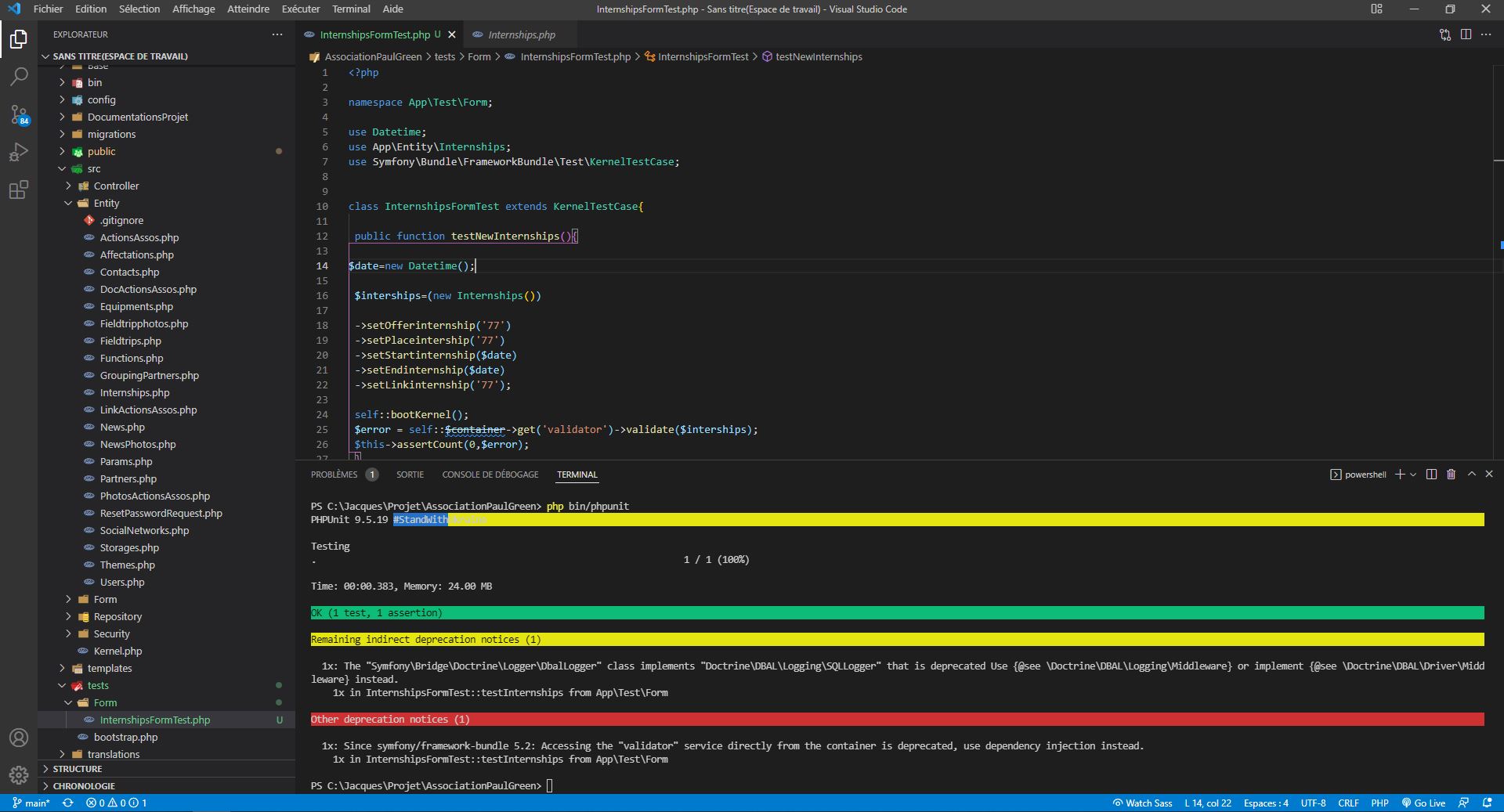
}

Pour lancer le test, il suffit d’exécuter la commande suivante :

PS C:\Jacques\Projet\AssociationPaulGreen> php bin/phpunit

dans notre cas les résultats sont Ok:





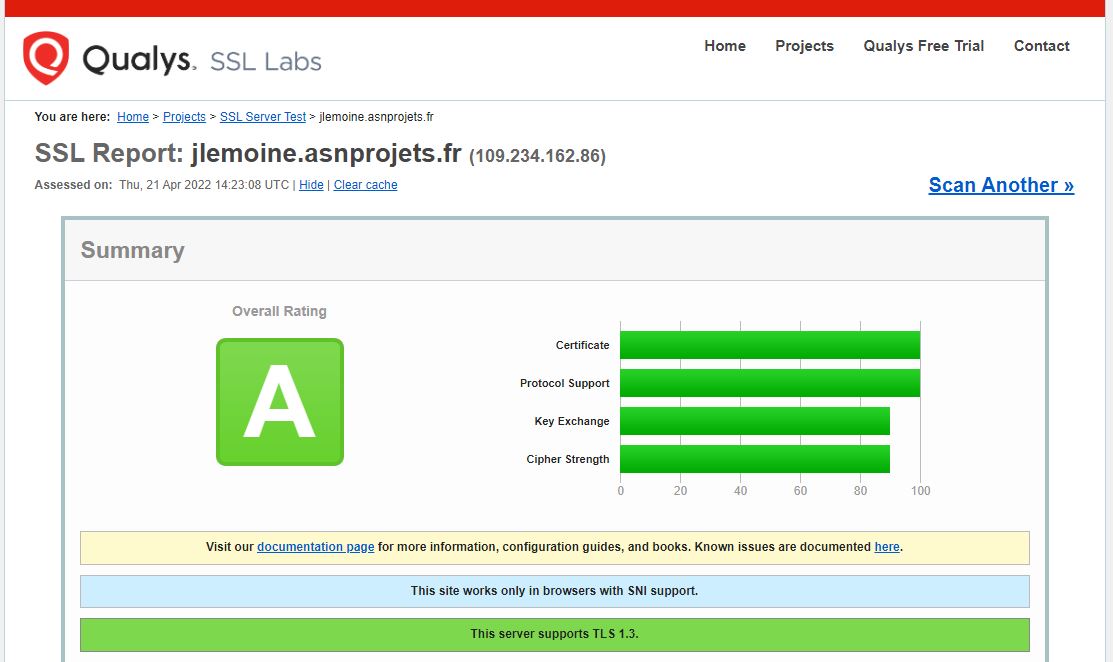
● **Mise en place du « https » et sécurité en production**

Le **https** ( = protocole de transfert hypertextuel sécurisé) est variante du protocole **http** (HyperText Transfer Protocol) chiffrée grâce au protocole **TLS** (Transport Layer Security). Il garantit la confidentialité et l'intégrité des données envoyées par et reçues du serveur.

Pour des raisons de **sécurité**, mais également pour optimiser **le référencement d’un site** (*Exemple : Dans ses résultats de recherche, Google favorise les sites en https*), il devient indispensable de mettre en place le https.

Pour cela, une fois le site **mis en production** à partir du panneau de contrôle **cPanel**, notre formateur utilise **Let’s Encrypt** qui permet de délivrer une certification de validation de domaine.

J’ai ensuite effectué des tests de sécurité grâce au site **SSL Labs** , qui a donné à mon site un score tout à fait satisfaisant.



● **Installation du reCAPTCHA**

Le **reCAPTCHA** protège votre site Web contre la fraude et les abus. Ce service de sécurité utilise un moteur d'analyse des risques avancés et des défis adaptatifs pour empêcher les logiciels malveillants de se livrer à des activités abusives sur les sites Web. Les utilisateurs légitimes pourront ainsi se connecter, effectuer des achats, afficher des pages ou créer des comptes tandis que les faux utilisateurs sont bloqués.

Afin de l’installer, je me suis inscrit sur le site<https://www.google.com/recaptcha/about/> et j’ai ensuite intégré le reCAPTCHA sur mon site via le code :

{# ============== reCAPTCHA ============== #}

<script src="https://www.google.com/recaptcha/api.js?render=6LdcH44fAAAAACrehdt6C6m6TerjtWj04krFK05w"></script>

<script>

function onClick(e) {

e.preventDefault();

grecaptcha.ready(function() {

grecaptcha.execute('6LdcH44fAAAAACrehdt6C6m6TerjtWj04krFK05w', {action: 'submit'}).then(function(token) {

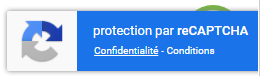
console.log('captacha');

});

});

}

</script>



Un symbole bleu apparaît alors en bas à droite de la page Web, signalant que le reCAPTCHA est actif.

J’ai ensuite fait un autre test qui vérifie la totalité des fichiers présents sur le site et détecte les malwares ainsi que les failles de sécurité éventuelles, il s’agit du site **SiteGuarding.com**. Il effectue aussi une vérification des liens présents sur le site en comparants ces derniers avec une liste de lien « blacklisté ». Aucun problème n’a été constaté lors de ce traitement dont voici un extrait.

## Front-End

### Mise en place du Framework CSS Bootstrap

Afin de développer plus rapidement la partie Front-End du site, ainsi que de profiter des fonctionnalités qu’il propose, j’ai choisi d’utiliser le framework Bootstrap.

Symfony est conçu plus précisément pour utiliser le framework **Bootstrap**.

##### Mise en place

Pour que les formulaires prennent le thème de Symfony avec les class= « form-control » et autres, il faut modifer le fichier de configuration de twig qui se trouve dans config/package/twig.yaml comme ci-dessous :

twig:

    default\_path: '%kernel.project\_dir%/templates'

    form\_themes: ['bootstrap\_5\_layout.html.twig']

when@test:

    twig:

        strict\_variables: true

Pour mettre en place ce framework, j’intègre les lignes suivantes dans le <head> du template base.html.twig comme ci-dessous :

    <!-- bootstrap-5.1.3-dist -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"

       rel="stylesheet"

       integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"

       crossorigin="anonymous">

puis ajouter le script JavaScript suivant avant la fermeture de la balise </body> :

<!-- Bootstrap  ==================================================== -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"

     integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"

     crossorigin="anonymous">

</script>

Mise en place des icons

J’ai aussi utilisé les icones de Font Awesome et pour cela j’ai également ajouté le

code JavaScript dans mon <head>

<!-- Icons-->

<script src="https://kit.fontawesome.com/4d2daebcf2.js"

crossorigin="anonymous"></script>

Mise en place des polices d’écritures

J’ai utilisé les fonts de Google Fonts, qui est un service d’hébergement gratuit de polices d’écritures pour le web.

Dans le <head> j’ajoute les lignes suivantes :

<head> les

codes:

   <!-- Fonts-->

   <link rel="preconnect" href="https://fonts.googleapis.com">

   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

   <link href="https://fonts.googleapis.com/css2?family=Gelasio:ital,wght@0,400;0,500;0,600;0,700;1,400;1,500;1,600;1,700&family=Parisienne&display=swap" rel="stylesheet">

Mise en place du CSS

J’ai décidé pour le CSS t’utiliser SASS, en partie pour la gestion plus simple du code

et de ses imbrications, son système @extend, et son système @import.

Pour la mise en place su SASS, il faut créer un fichier en .scss (styles.scss) et d’utiliser l’extension « Live Sass Compiler » dans Visual Studio Code. La compilation de ce fichier génère le fichier « styles.css ».

Je rappelle que le fichier CSS est lié avec la ligne suivante dans le <head> du fichier base.html.php :

        <!-- Styles -->

        <link rel="stylesheet" href="{{asset('css/styles.css')}}">

A titre d’information la fonction « asset » renvoi public, c’est-à-dire pour notre exemple

        <!-- Styles -->

        <link rel="stylesheet" href= "public/css/styles.css">

Le principe du SASS, est de pouvoir imbriquer les différentes commande CSS, plus calir pour la programmation.

// == FOOTER ============

footer {

    .clIconeEnLigne {

        display: flex;

        flex-direction: row;

        justify-content: center;

    }

    .clFooter {

        padding-top: 2rem;

        padding-bottom: 2rem;

        background-color: rgb(44, 62, 80);

        width: 100%;

        color: white;

    }

    .clCopyright {

        background-color: rgb(26, 37, 47);

        font-size: 70%;

        p {

            margin-top: 10px;

        }

    }

}

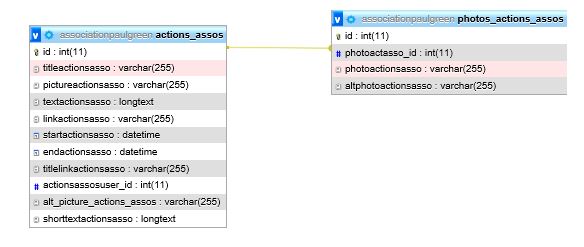
**Exemples de Codes**

Gestions de photos :

Dans la beaucoup de mes entités principales, le thème est exposé avec une photo principale mais les étudiants peuvent ajouter par la suite une multitude de photos rentrant dans des carousels.

Une méthode m’a été proposé par un tutoriel.

Pour notre exemple, nous allons utiliser la table ActionsAssos, et exposé la fonction de modification d’une action de l’association. Le principe est le même pour la partie création.



Dans notre base de données, la relation entre les deux tables est du type « One To Many ». dans la table regroupant les photos il sera possible ensuite de rajouter le texte alternatif indispensable pour les référencements. La relation entre les deux entités a été défini grâce à la commande « php bin/console make:entity » suivi des commande de migration précédemment.

Dans le cas ou les créations des fichiers de gestion générer par le CRUD ont déjà été réalisées, il faut modifier les différents fichiers « à la main ».

<?php

namespace App\Form;

use App\Entity\Themes;

use App\Entity\ActionsAssos;

use Symfony\Component\Form\AbstractType;

use FOS\CKEditorBundle\Form\Type\CKEditorType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Bridge\Doctrine\Form\Type\EntityType;

use Symfony\Component\Validator\Constraints\File;

use Symfony\Component\OptionsResolver\OptionsResolver;

use Symfony\Component\Form\Extension\Core\Type\DateType;

use Symfony\Component\Form\Extension\Core\Type\FileType;

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\Form\Extension\Core\Type\TextareaType;

class ActionsAssosType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

            ->add('titleactionsasso', TextType::class, [

                'label' => 'Titre : ',

                'required'=> true,

                'attr' => [

                    'maxlenght' =>'250',

                ]

            ])

            ->add('shorttextactionsasso', TextareaType::class, [

                'label' => 'Texte de présentation court : ',

                'required'=> false,

                'attr' => [

                    'rows' => '10',

                    'maxlength' => '1000'

                ],

            ])

            ->add('pictureactionsasso', FileType::class, [

                'label' => 'Photo de présentation: ',

                'mapped' => false,

                'required' => false,

                'constraints' => [

                    new File([

                        'maxSize' => '1024k',

                        'mimeTypes' => [

                            'image/\*',

                        ],

                        'mimeTypesMessage' => 'Veuillez entrer un format de document valide',

                    ])

                ],

            ])

            ->add('altPictureActionsAssos', TextType::class, [

                'label' => 'Texte alternatif Photo / sous-Titre : ',

                'required'=> false,

                'attr' => [

                    'maxlenght' =>'250',

                ]

            ])

            ->add('textactionsasso', CKEditorType::class, [

                'label' => 'Texte long : ',

            ])

            ->add('startactionsasso',DateType::class, [

                'label' => 'Date de début : ',

                'widget' => 'single\_text',

                'required' => false,

            ])

            ->add('endactionsasso',DateType::class, [

                'label' => 'Date de Fin : ',

                'widget' => 'single\_text',

                'required' => false,

            ])

            ->add('titlelinkactionsasso', TextType::class, [

                'label' => 'Titre du lien ci-dessous : ',

                'label\_attr' => [

                    'class' => 'fst-italic text-decoration-underline'

                ],

                'required'=> false,

                'attr' => [

                    'maxlenght' =>'250',

                ]

            ])

            ->add('linkactionsasso', TextType::class, [

                'label' => 'Lien : ',

                'label\_attr' => [

                    'class' => 'fst-italic'

                ],

                'required'=> false,

                'attr' => [

                    'maxlenght' =>'250',

                ]

            ])

            ->add('actionsassostheme', EntityType::class,[

                'label' => "Theme de l'action:",

                'class' => Themes::class,

                'choice\_label' => 'nameTheme',

                'multiple' => True,

                'expanded' => true,

                ])

            ->add('actionsassosphoto', FileType::class,[

                'label' => false,

                'multiple' => true,

                'mapped' => false,

                'required' => false,

                ])

        ;

    }

    public function configureOptions(OptionsResolver $resolver): void

    {

        $resolver->setDefaults([

            'data\_class' => ActionsAssos::class,

        ]);

    }

}

L’ajout des photos se fait par un select multiple de fichiers.

Les photos spécifiques à l’association sont enregistrées dans un répertoire particulier.

Cette information nous allons l’indiquer dans le fichier « config/services.yaml » dans la section « parameters ». Il suffit à chaque besoin pour afficher une photo d’utiliser le paramètre «photos\_directoryAssos ».

parameters:

    photos\_directoryAdherents: '%kernel.project\_dir%/public/photosAdherents'

    photos\_directoryParams: '%kernel.project\_dir%/public/photosParams'

    photos\_directoryPartners: '%kernel.project\_dir%/public/photosPartners'

    photos\_directoryNews: '%kernel.project\_dir%/public/photosNews'

    photos\_directoryAssos: '%kernel.project\_dir%/public/photosActionsAssos'

    photos\_directoryMasters: '%kernel.project\_dir%/public/photosActionsMasters'

    photos\_directoryFieldtrips: '%kernel.project\_dir%/public/photosFieldtrips'

Dans notre « controller » ActionsAssosControler.php, la méthode « edit » ci-dessous, et dans la partie des Photos multiples, nous contatons que la récupération des données se fait par un système de boucle avec « foreach », déplace la phot dans un répertoire spécifique et enregistre seulement un nouveau nom généré aléatoirement, dans la base de données.

    /\*\*

     \* @Route("/{id}/edit", name="actions\_assos\_edit", methods={"GET", "POST"})

     \*/

    public function edit(Request $request,

                                ActionsAssos $actionsAsso,

                                EntityManagerInterface $entityManager,

                                SluggerInterface $slugger,                                                         Users $users ): Response

    {

        $form = $this->createForm(ActionsAssosType::class, $actionsAsso);

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {

            // ================================= Photo

            $photo = $form->get('pictureactionsasso')->getData();

            if ($photo) {

                $originalFilename = pathinfo($photo->getClientOriginalName(), PATHINFO\_FILENAME);

                // ceci est nécessaire pour inclure en toute sécurité le nom de fichier dans l'URL

                $safeFilename = $slugger->slug($originalFilename);

                $newFilename = $safeFilename . '-' . uniqid() . '.' . $photo->guessExtension();

                // Déplacez le fichier dans le répertoire où les brochures sont stockées

                try {

                    $photo->move($this->getParameter('photos\_directoryAssos'), $newFilename);

                } catch (FileException $e) {

                    // ... gérer l'exception si quelque chose se produit pendant le téléchargement du fichier

                }

                // met à jour la propriété 'photo' pour stocker le nom du fichier au lieu de son contenu

                $actionsAsso->setPictureactionsasso($newFilename);

            }

            // ======================================

            // ========================== Photos Multiples

            // On récupère les images transmises

            $photos = $form->get('actionsassosphoto')->getData();

            // On boucle sur les images

            foreach($photos as $photo){

                // On génère un nouveau nom de fichier

                $fichier = md5(uniqid()).'.'.$photo->guessExtension();

                // On copie le fichier dans le dossier uploads

                $photo->move(

                    $this->getParameter('photos\_directoryAssos'),

                    $fichier

                );

                // On crée l'image dans la base de données

                $img = new PhotosActionsAssos();

                $img->setPhotoactionsasso($fichier);

                $actionsAsso->addActionsassosphoto($img);

            }

            // =====================================

            $actionsAsso->setActionsassosuser($this->getUser());

            $entityManager->flush();

            return $this->redirectToRoute('actions\_assos\_index', [], Response::HTTP\_SEE\_OTHER);

        }

// ----------------------

        // Partie commentaires

        // On crée le commentaire "vierge"

        $comment = new DocActionsAssos;

        // On génère le formulaire

        $commentForm = $this->createForm(DocActionsAssosType::class, $comment);

        $commentForm->handleRequest($request);

        // Traitement du formulaire

        if($commentForm->isSubmitted() && $commentForm->isValid()){

             // On récupère le contenu du champ parentid

             $parentid = $commentForm->get("docactasso")->getData();

             // On va chercher le commentaire correspondant

             if($parentid != null){

                 $parent = $entityManager

                    ->getRepository(DocActionsAssosType::class)

                    ->find($parentid);

             }

        return $this->render('actions\_assos/edit.html.twig', [

            'actions\_asso' => $actionsAsso,

            'form' => $form->createView(),

            'commentForm' => $commentForm->createView()

        ]);

    }

Pour supprimer les photos, j’ai utilisé une méthode « deleteImage » dans le même controller.

/\*\*

 \* @Route("/supprime/photo/{id}", name="actions\_assos\_delete\_photo", methods={"DELETE"})

 \*/

    public function deleteImage(Request $request, PhotosActionsAssos $photo, EntityManagerInterface $entityManager){

        $data = json\_decode($request->getContent(), true);

        // On vérifie si le token est valide

        if($this->isCsrfTokenValid('delete'.$photo->getId(), $data['\_token'])){

            // On récupère le nom de l'image

            $nom = $photo->getPhotoactionsasso();

            // On supprime le fichier

            unlink($this->getParameter('photos\_directoryAssos').'/'.$nom);

            // On supprime l'entrée de la base

            $entityManager->remove($photo);

            $entityManager->flush();

            // On répond en json

            return new JsonResponse(['success' => 1]);

        }else{

            return new JsonResponse(['error' => 'Token Invalide'], 400);

        }

    }

J’associe à cette méthode du JavaScript couplé avec une requête Ajax, avec l’appui sur un bouton. J’ai créé un fichier photos.js que j’ai intégrer dans mon projet avec le code suivant :

window.onload = () => {

    // Gestion des boutons "Supprimer"

    let links = document.querySelectorAll("[data-delete]")

    // On boucle sur links

    for (link of links) {

        // On écoute le clic

        link.addEventListener("click", function(e) {

            // On empêche la navigation

            e.preventDefault()

            // On demande confirmation

            if (confirm("Voulez-vous supprimer cette photo ?")) {

                // On envoie une requête Ajax vers le href du lien avec la méthode DELETE

                fetch(this.getAttribute("href"), {

                    method: "DELETE",

                    headers: {

                        "X-Requested-With": "XMLHttpRequest",

                        "Content-Type": "application/json"

                    },

                    body: JSON.stringify({ "\_token": this.dataset.token })

                }).then(

                    // On récupère la réponse en JSON

                    response => response.json()

                ).then(data => {

                    if (data.success)

                        this.parentElement.remove()

                    else

                        alert(data.error)

                }).catch(e => alert(e))

            }

        })

    }

}

J’ai intégrer le lien à ce fichier avec une balise script.

    <script src="{{asset('js/photos.js')}}" defer></script>

Dans tous les fichiers templates Twig, ou je gére ces photos j’intégre le code par exemple suivant, qui m’affiche chaque photo avec le bouton supprimer :

<p>Photos</p>

<div class="d-flex flex-row">

    {% for photo in actions\_asso.actionsassosphoto %}

        <div class="card me-2" style="width: 150px;">

            <img class="card-img-top" src="{{ asset('/photosActionsAssos/'~photo.photoactionsasso) }}"

                alt="{{ photo.altphotoactionsasso }}" width="150" title="{{ photo.photoactionsasso }}">

            <div class="card-body p-0 d-flex flex-column justify-content-between">

                <p class="card-text textAlt">{{ photo.altphotoactionsasso }}</p>

                <div class="card-footer">

                    <a href="{{ path('actions\_assos\_delete\_photo', {id: photo.id})}}"

                        data-delete data-token="{{ csrf\_token('delete' ~ photo.id )}}"

                        class="btn btn-grad btn-danger2">

                        Supprimer

                    </a>

                </div>

            </div>

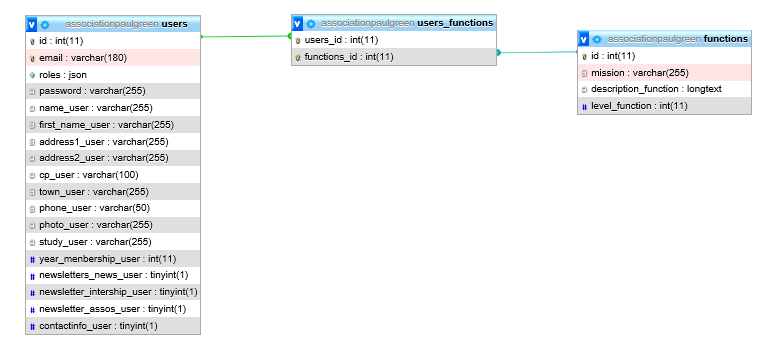
        </div>

    {% endfor %}

</div>

Affichage du Bureau :

Le but ici est d’afficher un bureau, c’est à dire les Users qui ont une fonction au sein de l’association et suivant un ordre précis précisé dans l’entité Functions.



La relation entre les tables users et functions est une « Many To Many », d’où la table pivot users\_functions.

J’ai recherché à traduire une requete SQL classique, réalisé avec la partie SQL dans phpMyAdmin, dans le système de requete Symfony. J’ai donc réalisé une function dans le repository UsersRepository.php.

// SELECT users.\*, function.\* FROM users

// INNER JOIN users\_functions ON users.id=users\_functions.users\_id

// INNER JOIN functions ON users\_functions.functions\_id=functions.id

// ORDER BY functions.level\_function, users.name\_user, users.first\_name\_user

public function findOffice()

    {

            $entityManager = $this->getEntityManager();

            $query = $entityManager->createQuery(

                    'SELECT u, f

                     FROM App\Entity\Users u

                     INNER JOIN u.functionUser f

                     ORDER BY f.levelFunction, u.nameUser, u.firstNameUser'

            );

            return $query->getResult();

}

Dans le controller, je fais appel à cette fonction pour l’affichage du bureau.

    /\*\*

     \* @Route("/Office", name="office\_index", methods={"GET"})

     \*/

    public function office(UsersRepository $usersRepository,

                            ParamsRepository $paramsRepository,

                            SocialNetworksRepository $socialNetworksRepository ): Response

    {

        return $this->render('users/office.html.twig', [

            'office' => $usersRepository->findOffice(),

            'params' => $paramsRepository->findAll(),

            'social\_networks' => $socialNetworksRepository->findAll(),

        ]);

    }

Puis j’ai créé un Template pour afficher les membres du bureau avec ce code ci-dessous :

<div class="container clBureau">

    <h1 class="h-grad bg-blueGreen">Les membres du Bureau</h1>

        <hr class="mb-3">

        <div class="row row-cols-1 row-cols-md-3 g-4">

            {% for user in office %}

                <div class="col">

                    <div class="card text-center ombre">

                        <div class="d-flex justify-content-center">

                            {% if user.photoUser %}

                                <img  class="border-end border-bottom border-5 rounded-circle"

                                    src="{{ asset('photosAdherents/' ~ user.photoUser) }}"

                                    alt="{{ user.nameUser }} {{ user.firstNameUser }}">

                            {% else %}

                                <img  class="border-end border-bottom border-5 rounded-circle"

                                    src="{{ asset('icones/PasDeTete.png') }}"

                                    alt="{{ user.nameUser }} {{ user.firstNameUser }}">

                            {% endif %}

                        </div>

                        <div class="card-body d-flex flex-column align-items-center">

                            <h5 class="card-title">{{ user.nameUser }} {{ user.firstNameUser }}</h5>

                            <a class="clRond"

                                href="mailto:{{ user.email }}

                                    ?subject=Sujet du mail...

                                    &cc={{ params[0].emailParam}}

                                    &body=Bonjour {{ user.nameUser }} {{ user.firstNameUser }} ...

                                    "

                                    rel="" title="">

                                <i class="fas fa-envelope"></i>

                            </a>

                            <a class="card-text text-decoration-none"

                                href="mailto:{{ user.email }}

                                    ?subject=Sujet du mail...

                                    &cc={{ params[0].emailParam}}

                                    &body=Bonjour {{ user.nameUser }} {{ user.firstNameUser }} ...

                                    "

                                    rel="" title="">

                                <i class="fas fa-envelope"></i> {{ user.email }}

                            </a>

                            <br>

                            <a href="tel:+{{ user.phoneUser }}" class="card-text text-decoration-none">

                                <i class="fas fa-phone-square-alt"></i> {{ user.phoneUser }}

                            </a>

                        </div>

                        <div class="card-footer">

                            {% set xFonction = null %}

                            {% for xFct in user.functionUser %}

                                {% if xFonction is null %}

                                    {% set xFonction = xFct.mission %}

                                {% else %}

                                    {% set xFonction = xFonction ~ ' / ' ~ xFct.mission %}

                                {% endif %}

                            {% endfor %}

                            <small class="text-muted">{{ xFonction }}</small>

                        </div>

                    </div>

                </div>

            {% else %}

                <p>Aucun membre du Bureau</p>

            {% endfor %}

        </div>

        <hr class="mb-3">

        <a class="btn btn-grad btn-secondary2 me-2" href="{{ path('users\_index') }}">Retour</a>

        <hr class="mb-3">

    </div>

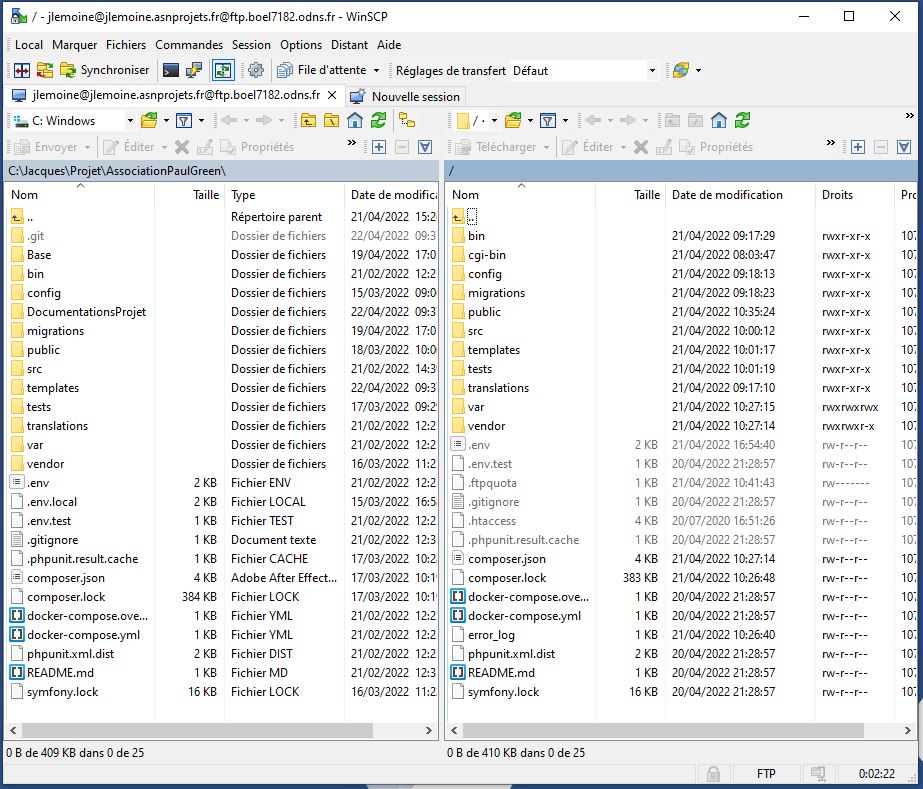
## Mise en production

### Hébergement

Le projet étant toujours en développement, pour les besoins de l’examen, le site a été installé sur Arinfo par notre formateur.

**O2Switch** est l’hébergeur français utilisé par Arinfo.

Afin de transférer les données locales de l’ordinateur, le client SFTP (SSH File Transfer Protocol) a été utilisé.

**WinSCP.**

Après avoir importé la base données via cPanel, le fichier ‘.env’  est modifié pour la nouvelle base :

DATABASE\_URL=mysql://boel7182:'fpHB4x-DxUJI'@109.234.162.86:3306/boel7182\_jlemoine

Le fichier ‘.htaccess’ a été ajouté à la racine de mon dossier public afin de gérer les redirections.

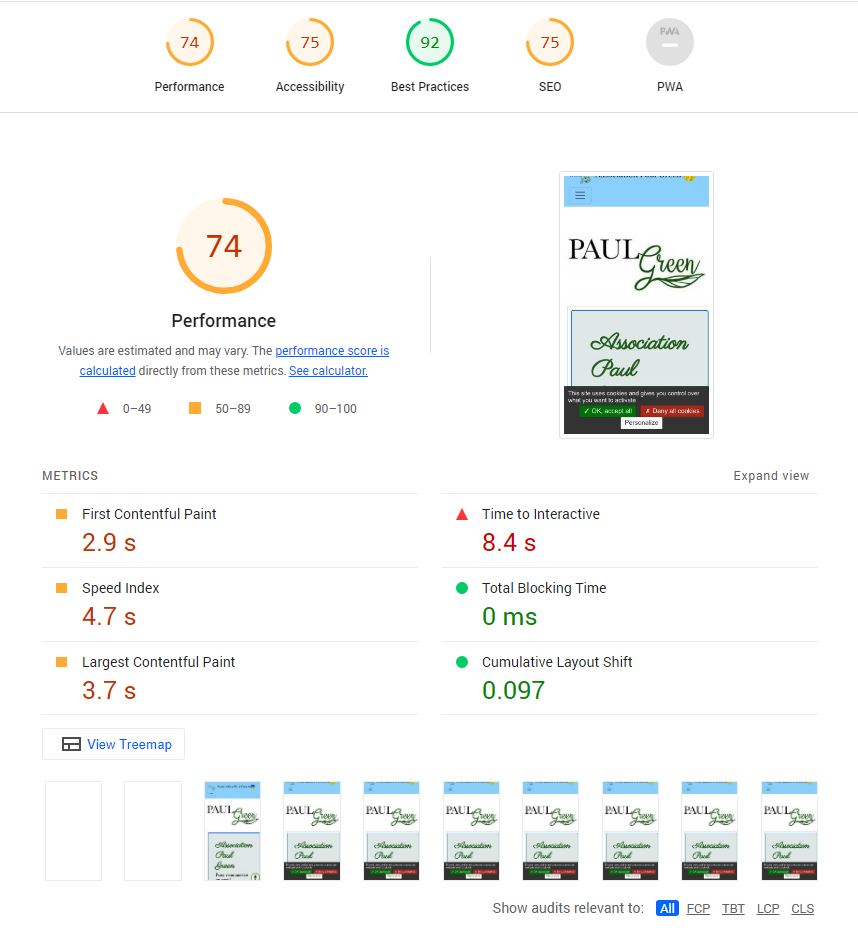
### 

### Optimisation des performances et Référencement

##### Lighthouse

Afin d’évaluer les performances du site internet, j’ai utilisé l’outil Google Lighthouse, qui permet de mesurer la qualité des pages web.

Avec le premier test, nous avons les résultats suivants :



Il est important de lancer des axes de progrès du site afin d’améliorer ces évaluations.

* **SEO (Search Engine Optimisation) ou Optimisation pour les moteurs de recherche.**

Pour améliorer le référencement, j’ai complété les informations de métadonnées dans le **‘head’** de ma page base.html.twig.

<!DOCTYPE html>

<html lang="fr">

<html>

    <head>

        <meta charset="UTF-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <meta name="description"

            content={{render(controller('App\\Controller\\MainController::paramRecup'))}} &

                " || Campus BIODIV+  ||  Faire de l’Université Paul Valéry un campus à biodiversité positive" &

                " || Université Paul Valéry Montpellier 3 || Master Gestion de l'Environnement"

        <title>

            {% block title %}

                {{render(controller('App\\Controller\\MainController::paramRecup'))}}

            {% endblock %}

        </title>

J’ai également pris soin de remplir l’attribut de texte alternatif ‘alt’ de l’ensemble des images du site. En sachant que les images rentrées par les utilisateurs pour les différentes entités, il y a la possibilité de saisir eux même les textes alternatifs.

En exemple :

<div class="d-flex justify-content-center">

<img src="{{ asset('photosParams/' ~ params[0].photoParam) }}"

           alt="{{ params[0].titleParam }}"

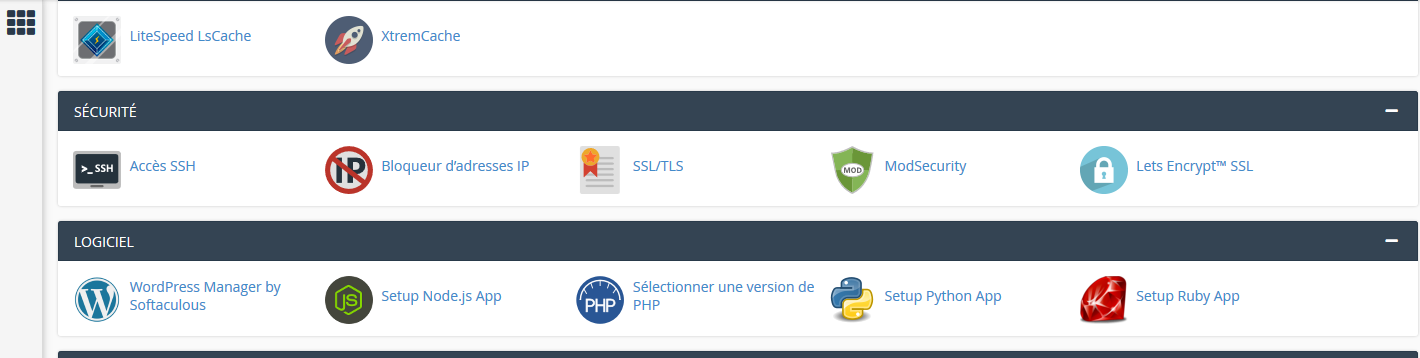
loading="lazy"

           width="250" height="150" class="imgClass rounded-3">

   </div>

* **Bonnes Pratiques**

A partir du panneau de configuration de cPanel, pour augmenter la sécurité du site, et améliorer son référencement, l’application est configuré avec le protocole **‘https’.**





### 

### Cookies/RGPD

##### Tarte au citron

Pour pouvoir gérer les cookies sur mon site, j’ai mis en place le système de gestion appelé ‘Tarte au citron’.

Après avoir téléchargé les fichiers nécessaires et les avoir inclus dans mon projet, j’ai copié ces scripts qui me permettent notamment de gérer les options de ce système :

{# ============== Tarteaucitron ============== #}

    <script type="text/javascript" src="{{ asset('tarteaucitron/tarteaucitron.js')}}"></script>

    <script>

        tarteaucitron.init({

            "privacyUrl": "", /\* URL de la page de la politique de vie privée \*/

            "hashtag": "#tarteaucitron", /\* Ouvrir le panneau contenant ce hashtag \*/

            "cookieName": "tarteaucitron", /\* Nom du Cookie \*/

            "orientation": "bottom", /\* Position de la banniére (top - bottom) \*/

            "showAlertSmall": false, /\* Voir la bannière réduite en bas à droite \*/

            "cookieslist": true, /\* Voir la liste des cookies \*/

            "adblocker": false, /\* Voir une alerte si un bloqueur de publicités est détecté \*/

            "AcceptAllCta": true, /\* Voir le bouton accepter tout (quand highPrivacy est à true) \*/

            "highPrivacy": true, /\* Désactiver le consentement automatique \*/

            "handleBrowserDNTRequest": false, /\* Si la protection du suivi du navigateur est activée, tout interdire \*/

            "removeCredit": false, /\* Retirer le lien vers tarteaucitron.js \*/

            "moreInfoLink": true, /\* Afficher le lien 'voir plus d'infos' \*/

            "useExternalCss": false, /\* Si false, tarteaucitron.css sera chargé\*/

            //"cookieDomain": '.my-multisite-domaine.fr', /\* Cookie multisite \*/

            "readmoreLink": "", // '/cookiespolicy' /\* Lien vers la page 'Lire plus' \*/

            // =====

            "showIcon": false, /\* Show cookie icon to manage cookies \*/

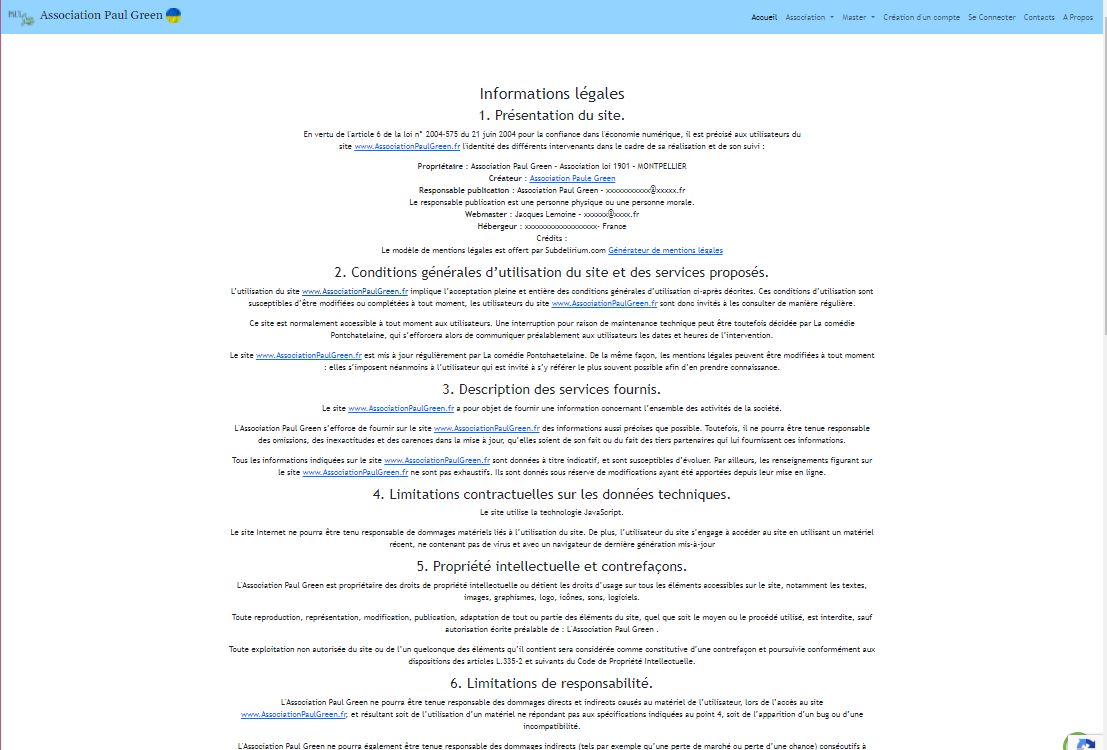
        });

    </script>



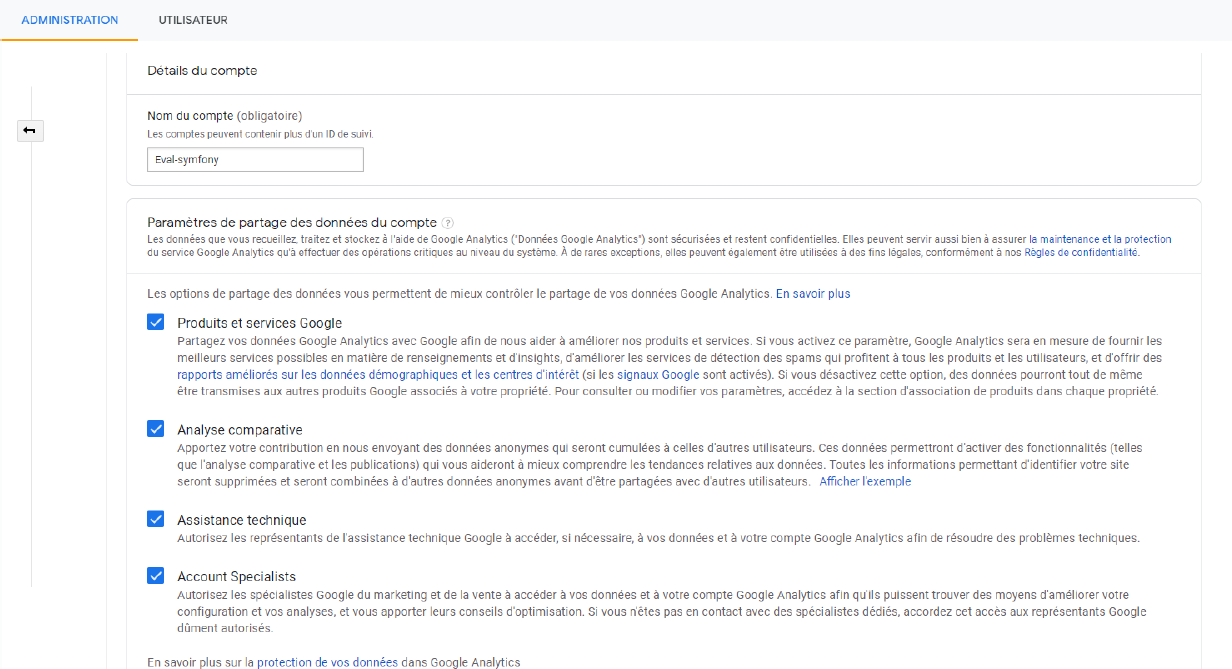
Ceci permet d’afficher la bandeau sur les cookies au démarrage du site.

##### RGPD

Depuis le **bas de page du site (Footer**), l’utilisateur peut accéder à une page de **mentions légales. Celle-ci est entièrement modifiable et adaptable par les administrateurs.**

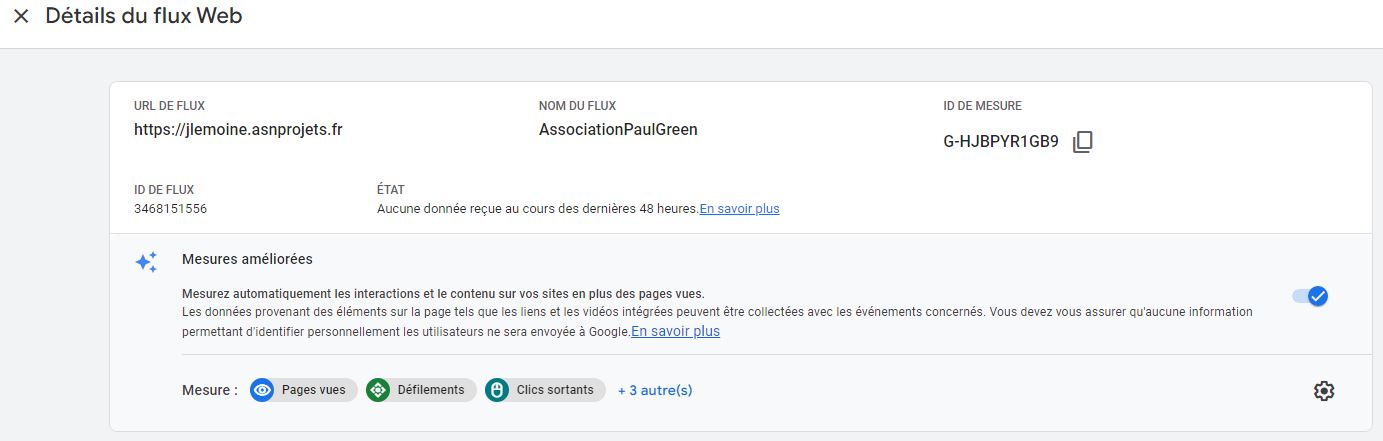


**Intégration de Google Analytics**



Google Analytics est un service gratuit d'analyse d'audience d'un site Web ou d'applications.

Pour profiter de ce service, il faut posséder un compte Google et s’inscrire le site Projet sur Google Analytics.

Cette inscription est suivie d’une phase de configuration, en fonction des objectifs du site internet et de l’identité du destinataire du site.  
Une fois toutes ces étapes franchies, il faut intégrer le code d’identification.

Pour finaliser la configuration il faut intégrer le code ci-dessous dans le <head>.

{# ============== Global site tag (gtag.js) - Google Analytics ============== #}

    <script async src="https://www.googletagmanager.com/gtag/js?id=G-HJBPYR1GB9"></script>

    <script>

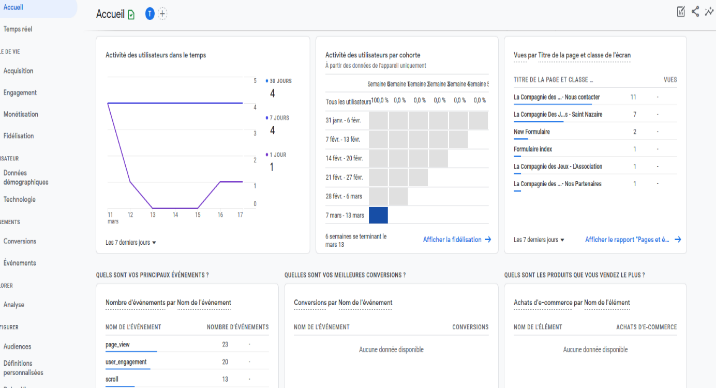
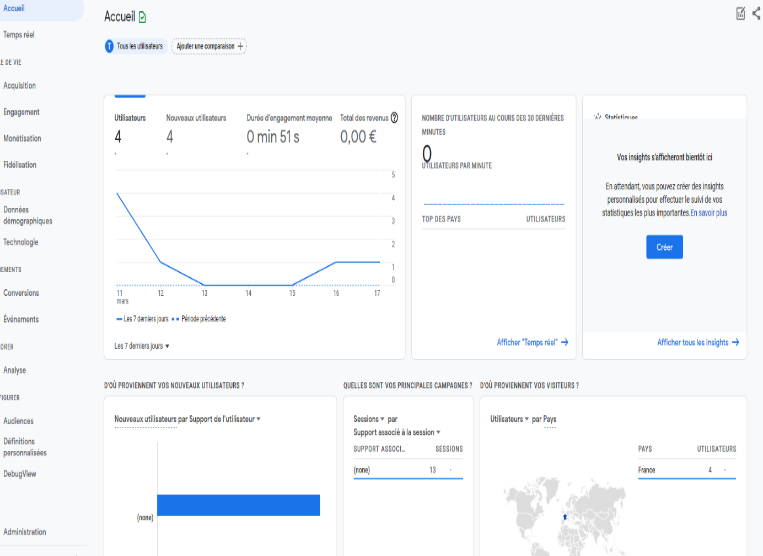
        window.dataLayer = window.dataLayer || [];

        function gtag(){dataLayer.push(arguments);}

        gtag('js', new Date());

        gtag('config', 'G-HJBPYR1GB9');

    </script>

Google Analytics nous permet alors, via un tableau de bord personnalisable, de consulter et de pouvoir analyser l’activité de notre site.

Le site n’étant en ligne depuis très peu de temps les résultats sont bien sûr non significatifs.