

**DOSSIER PROJET**

L’association **PAUL** **Green**



**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.

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

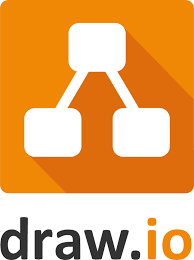
Le projet et son déroulement furent très appréciés par le bureau de l’association des étudiants sachant que ce site pour répondre complètement aux besoins de l’association qui ne cessent d’évoluer, devra être développé davantage, ce que je pourrai prétendre une fois ma formation finie afin de leur proposer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

**Contraintes techniques :**

Il n’y a pas de nom de domaine ni de serveur existant.

Choix du nom de domaine: www……………………………………..

Choix de l’hébergeur : ………………………………

Veille

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.



Trebuchet





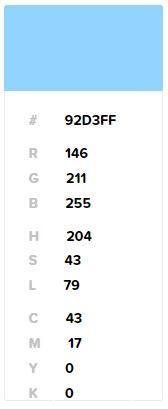
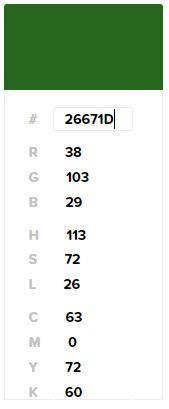
Gelasio



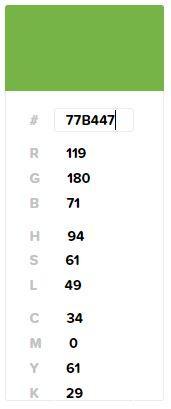
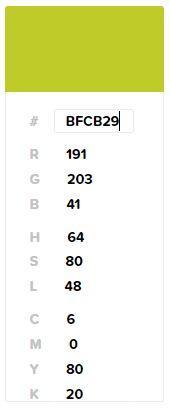
Parisienne

**Pour le choix des couleurs :**

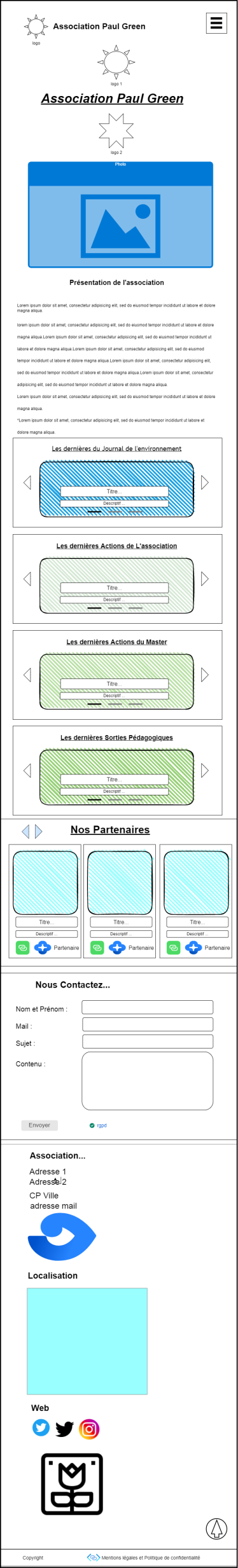
Le chemin directeur pour le choix des couleurs est également parti du découpage du logo fourni par l’association.

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

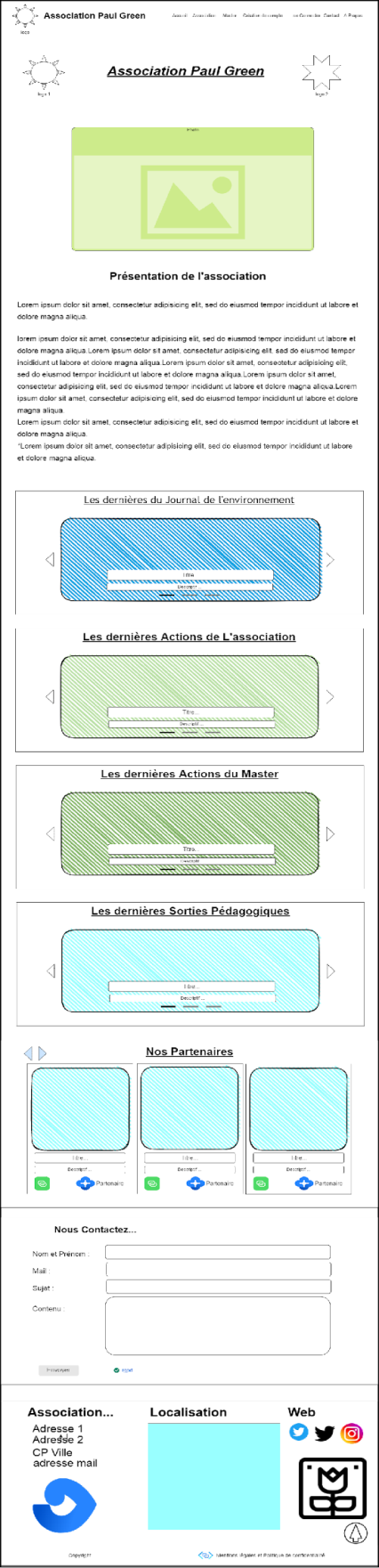




Wireframe et Maquettage 



**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.

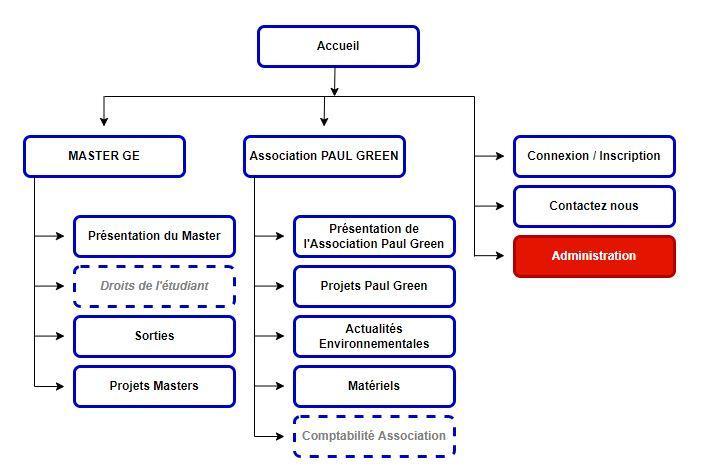
À droite nous avons la wireframe de la page d’accueil responsive.

.

Arborescence du site

**Voici l’arborescence du site:**

A REPRENDRE – revoir le menu



**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 derniers actions menées par l’association
* présentation des 5 derniers 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 masters, de gérer complètement les utilisateurs, les différentes fiches actions associatives, du masters, 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 Diagrammes Uml

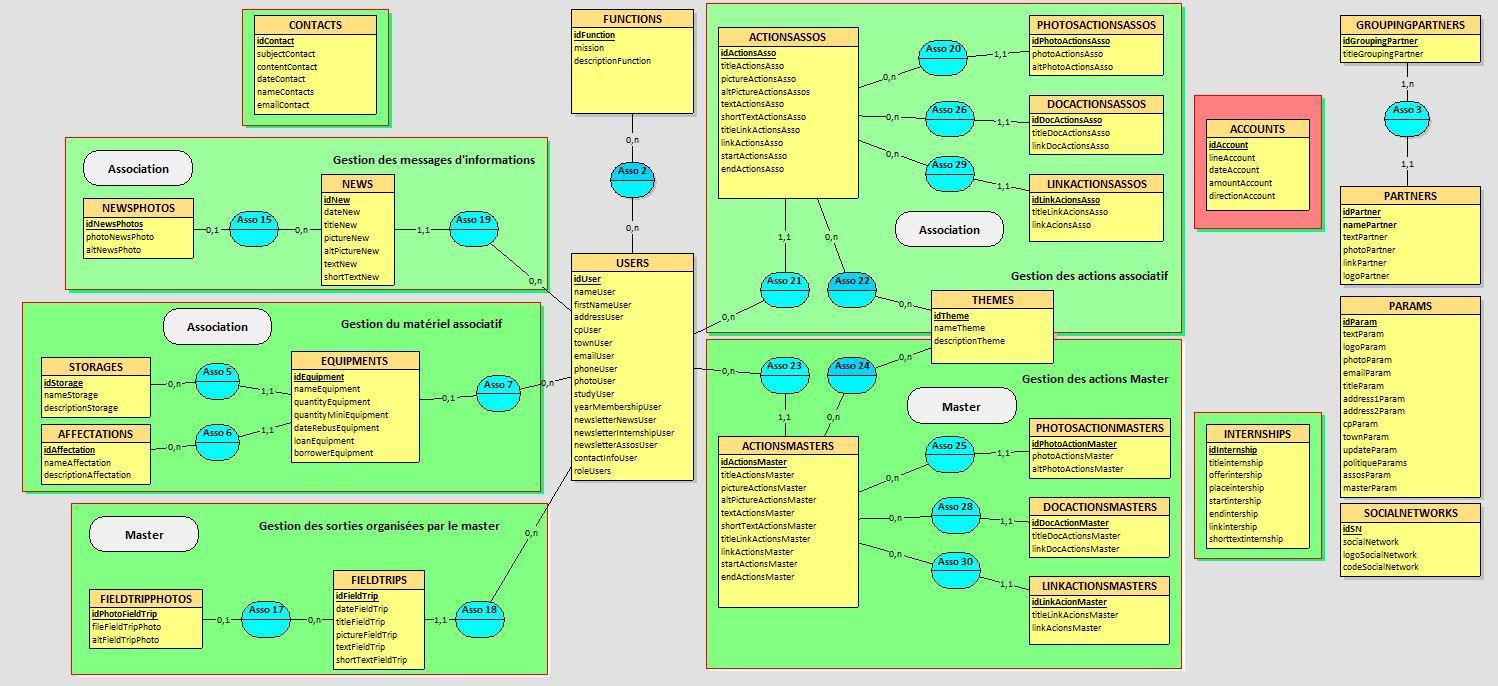


Afin de construire les **MCD (Modèles Conceptuels de Données)** et les **MLD (Modèles Logiques de Données**, j’ai utilisé le logiciel **Looping**. Ce logiciel est gratuit et libre d’utilisation. Il permet la modélisation des entités avec leurs propriétés et types de données, les associations et cardinalités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

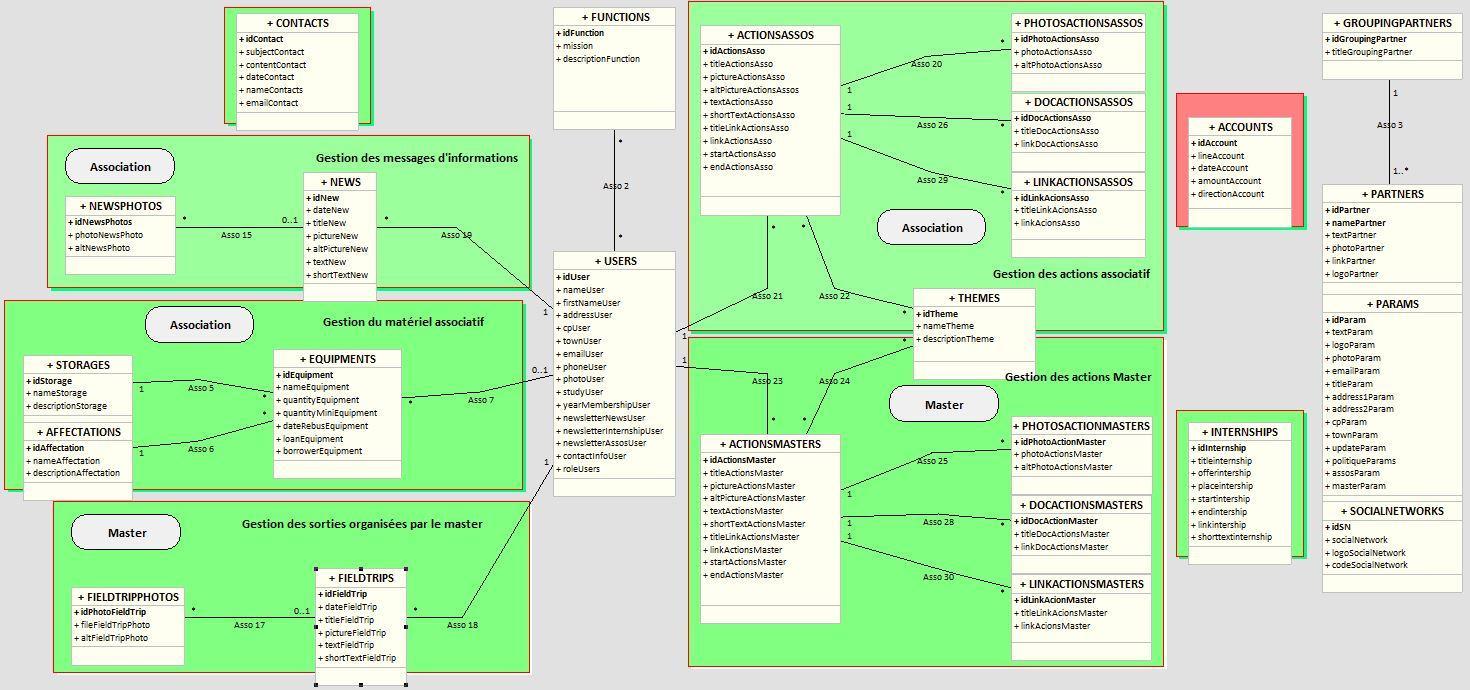
Les entités sont en Majuscule et les propriétés sont en camel case, 1er mot tout en minuscule et les suivants attachés, en minuscule avec l'initiale en majuscule.



**UML**

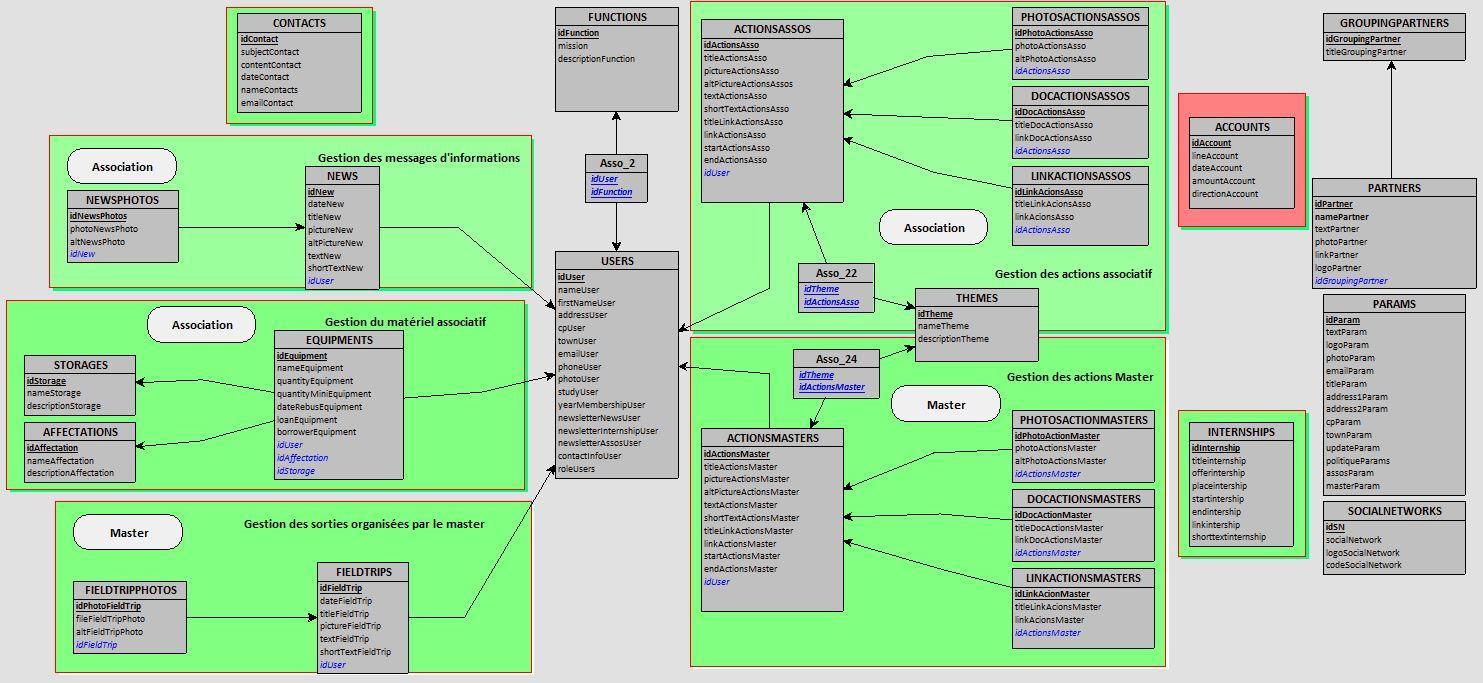


L’ UML ( Langage de Modélisation Unifié ) est généré automatiquement à partir du MCD réalisé.



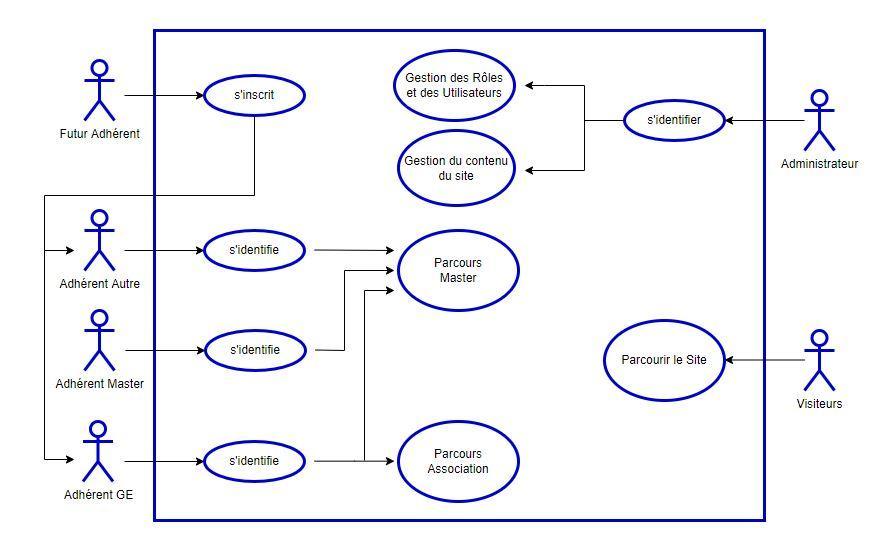
**MLD**

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



**Use Case Diagramme**

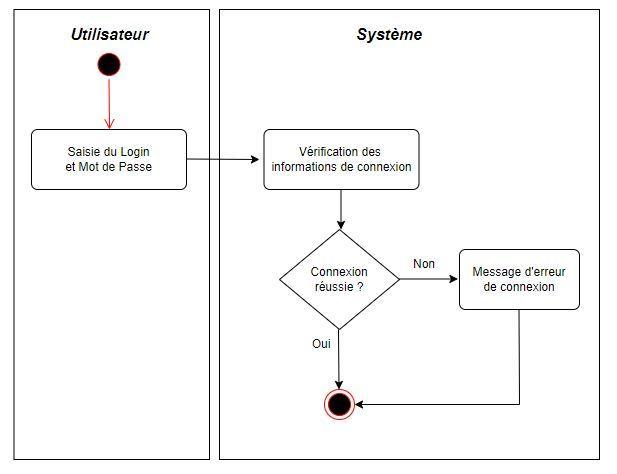
Afin d’avoir une vision globale du comportement fonctionnel du site en fonction de chaque rôle utilisateur, je réalise un diagramme de cas d’utilisation.



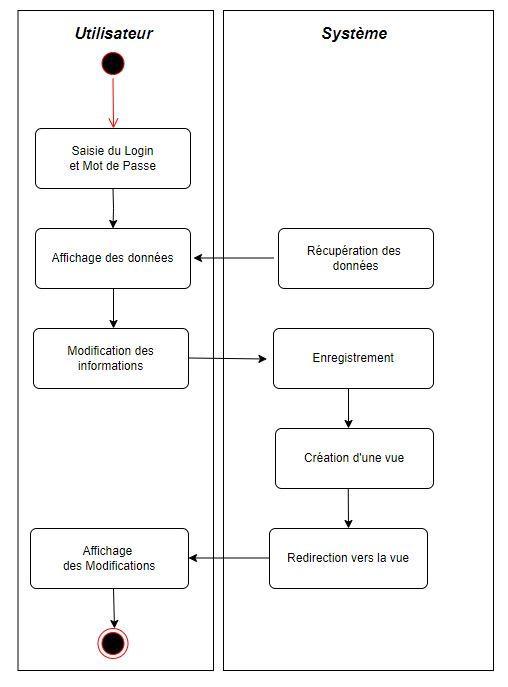
**Diagrammes d’activités**

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’activité connexion utilisateur**



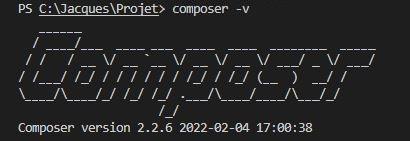
**Diagramme d’activité modification de contenu**



Développement Back-End

**CRÉATION De la structure de la base données 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…

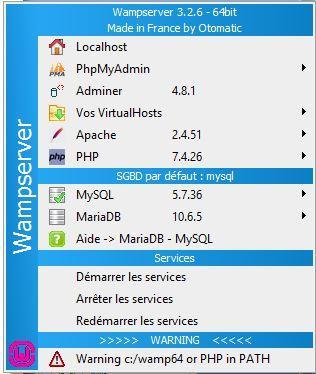


La ligne de commande suivante, grâce au logiciel Composer, installe Symfony, notre socle pour notre projet. Il crée le répertoire « AssociationPaulGreen », avec l’ensemble des modules pour Symfony.

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 ###

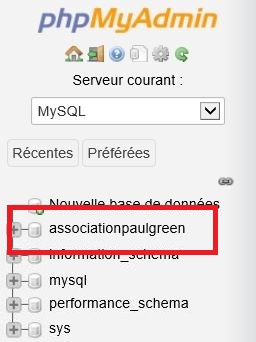
DATABASE\_URL=mysql://root:@127.0.0.1:3306/AssociationPaulGreen?serverVersion=5.7&charset=utf8mb4

###< 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 D’UNE 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.

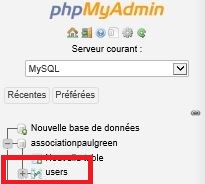
Les lignes de commande dans le terminal suivantes lancent la mise à jour de la base données.

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 login 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

**CONSTRUCTION DU FORMULAIRE D’ENREGISTREMENT**

La création de ce formulaire de saisie se fait d’une manière automatique par l’intermédaire 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 tape l’adresse  « /equipments » directement sans avoir les droits admin il serait renvoyé 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 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 Article :

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!

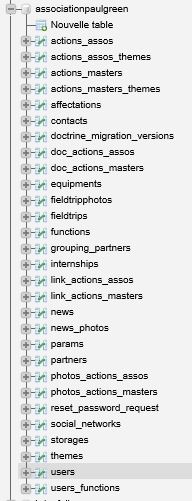
Next: When you're ready, create a migration with make:migration

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.

arborescence

Suivant les niveaux d’accès certaine option sont désactivé

Dans le cas d’une connection en administrateur, la partie administrateur s’affiche

avec cette architecture en plus de la plage d’accueil

mettre l’arborescence 2

arborescence Administrateur

—---------------------------------------------------

##### 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 biais.

##### **Sécurisation et tests**

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.

J’ai également consulté le blog de [Vaadata](https://www.vaadata.com/fr/), une société spécialisée dans le ‘pentest’ ( = test d’intrusion). J’ai donc pris quelques mesures afin de rendre mon site plus sûr.

● **Tests unitaires**

Pour effectuer des tests unitaires j’ai installé le framework opensource PHPUnit.

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/browse-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);

}

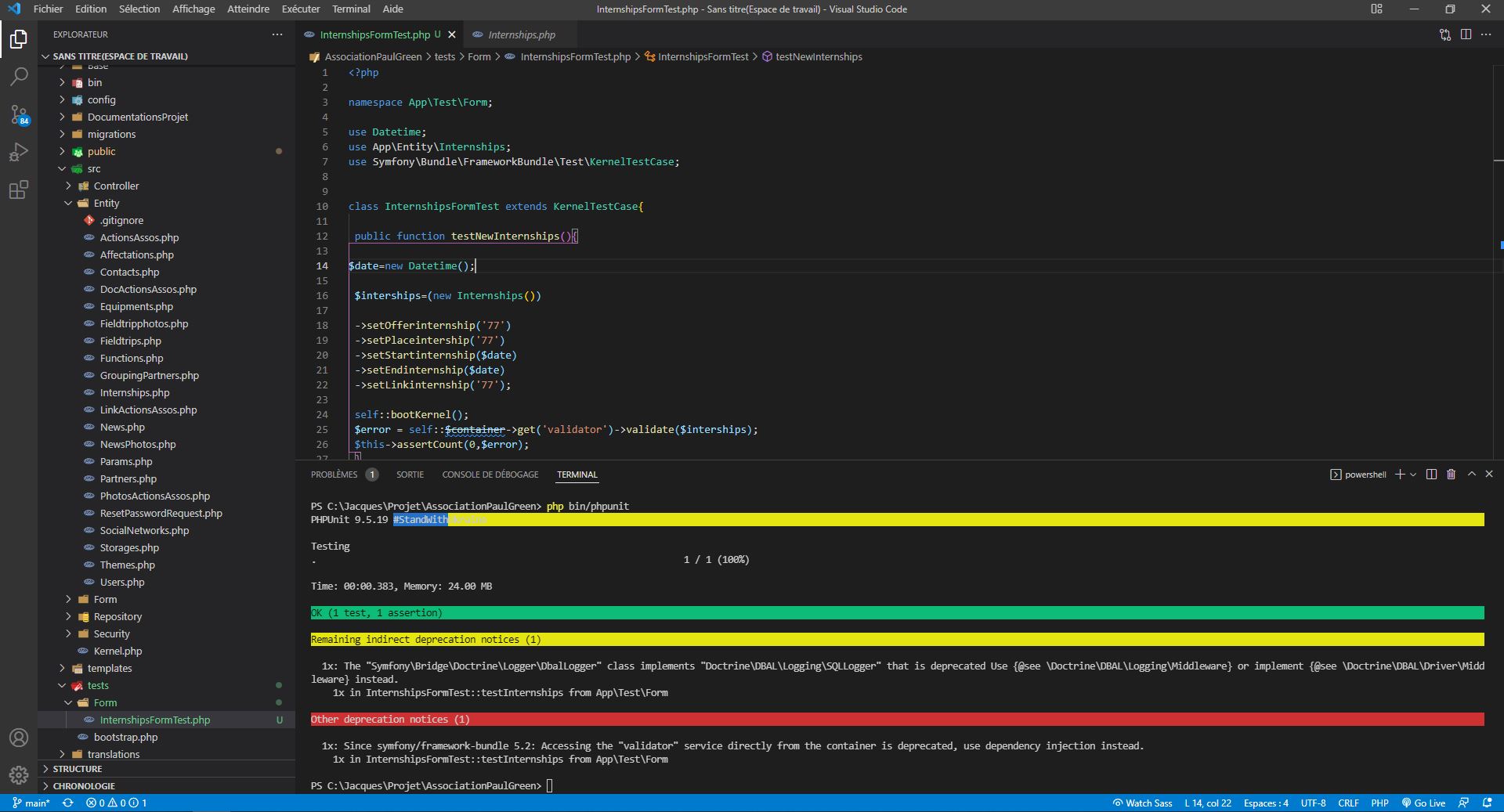
}

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:





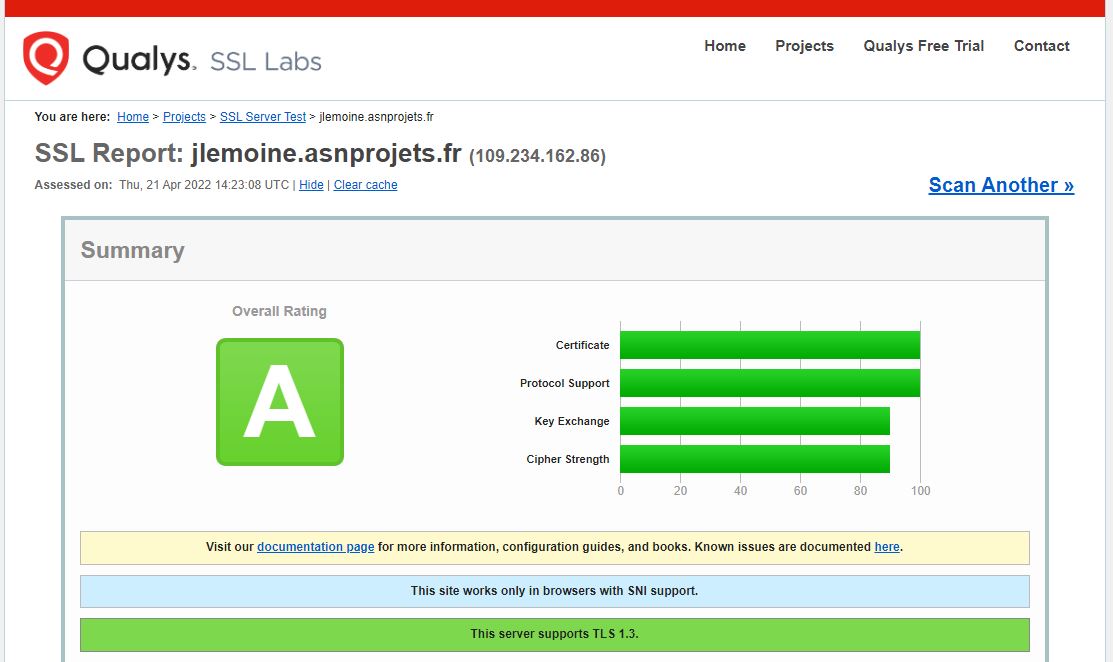
● **Mise en place du « https » et sécurité en production**

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) {

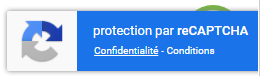
console.log('captacha');

});

});

}

</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 ».

Ma démarche est assez simple je crée un nouveau fichier .scss pour chaque grande

partie de mon site et j’ai donc un fichier style.scss pour la partie «Annuaire», et un

fichier styleAD.scss pour la partie «Administration» du site.

Sur Visual studio code j’ai installé une extension qui me permet de compiler mon

fichier .scss en un fichier .css.

// NAV

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

nav {

border-bottom: $brown solid 1px;

li {

display: inline-block;

min-width: 50px;

a {

text-decoration: none;

line-height: 30px;

padding: 0 10px;

text-align: center;

font-size: 1rem;

font-weight: 600;

color: $brown !important;

}

}

}

Je trouve

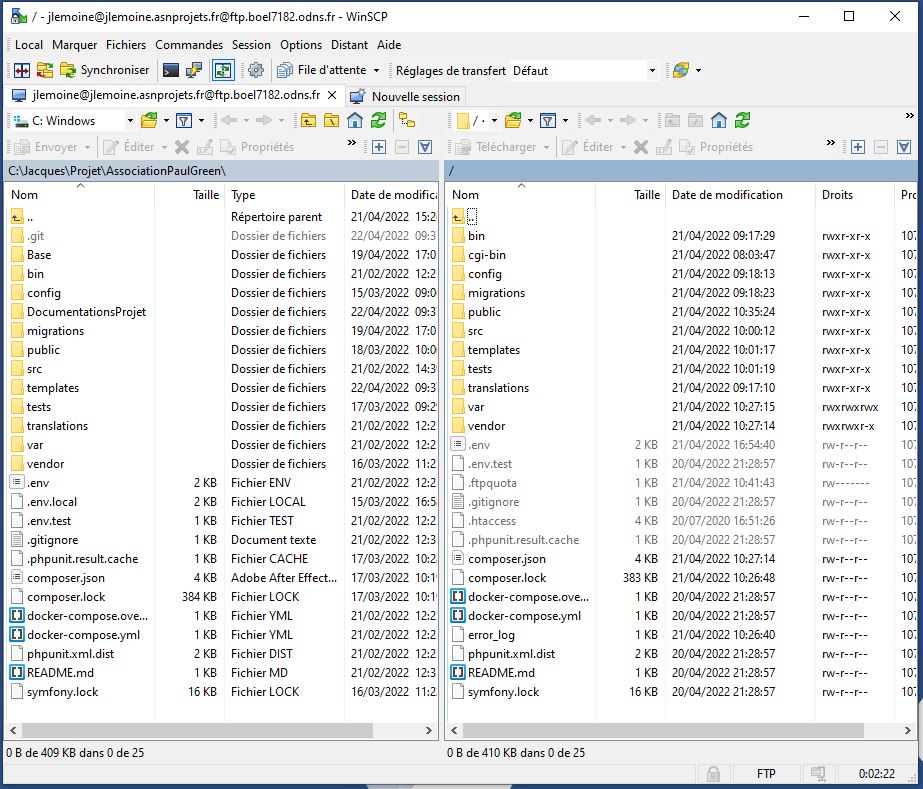
## Mise en production

### 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.

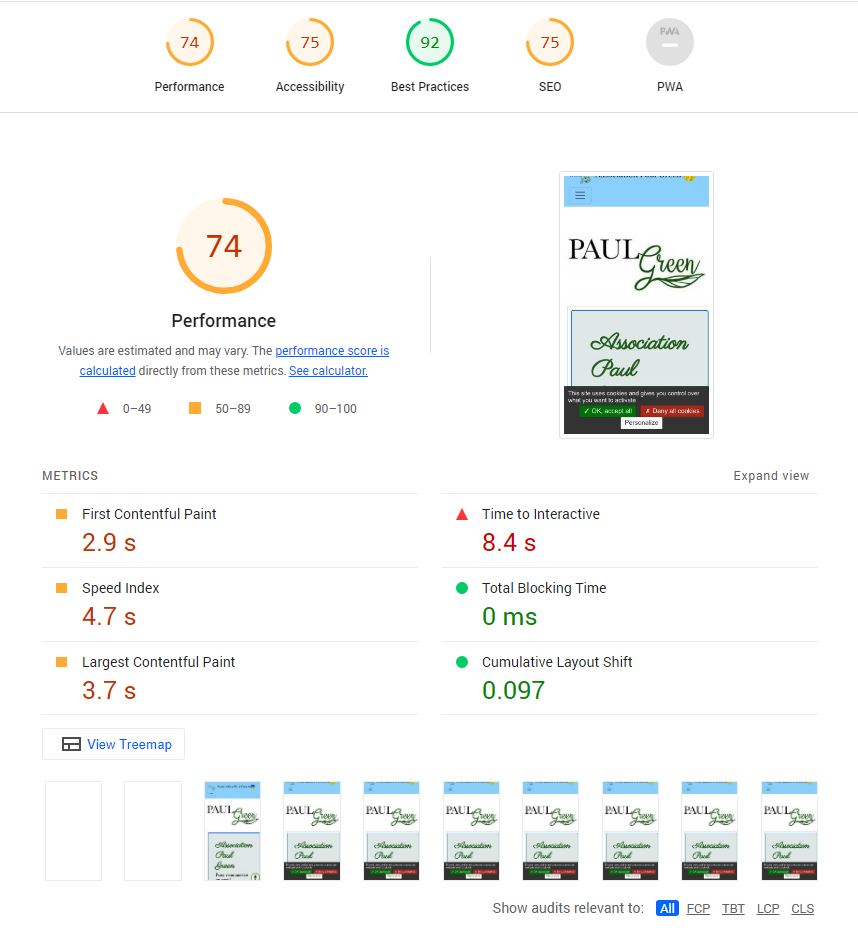
### 

### 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 }}"

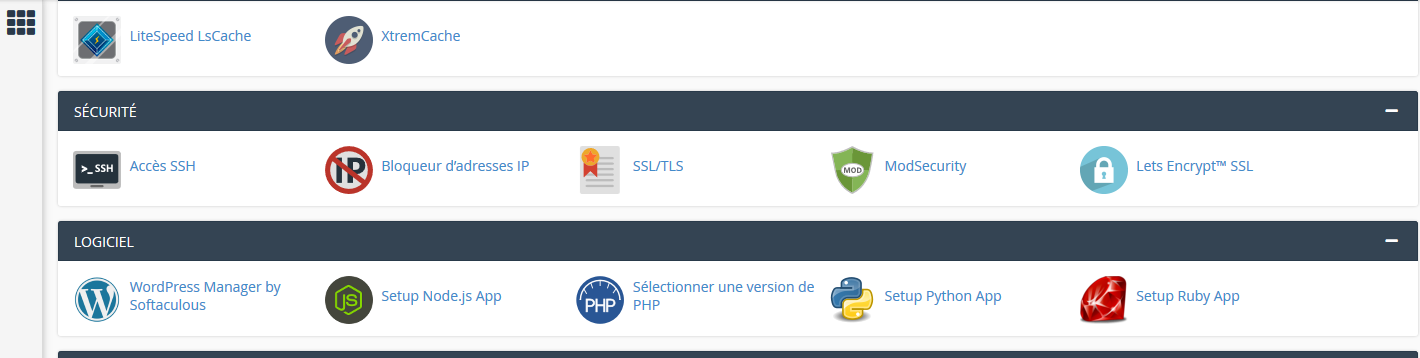
loading="lazy"

           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' \*/

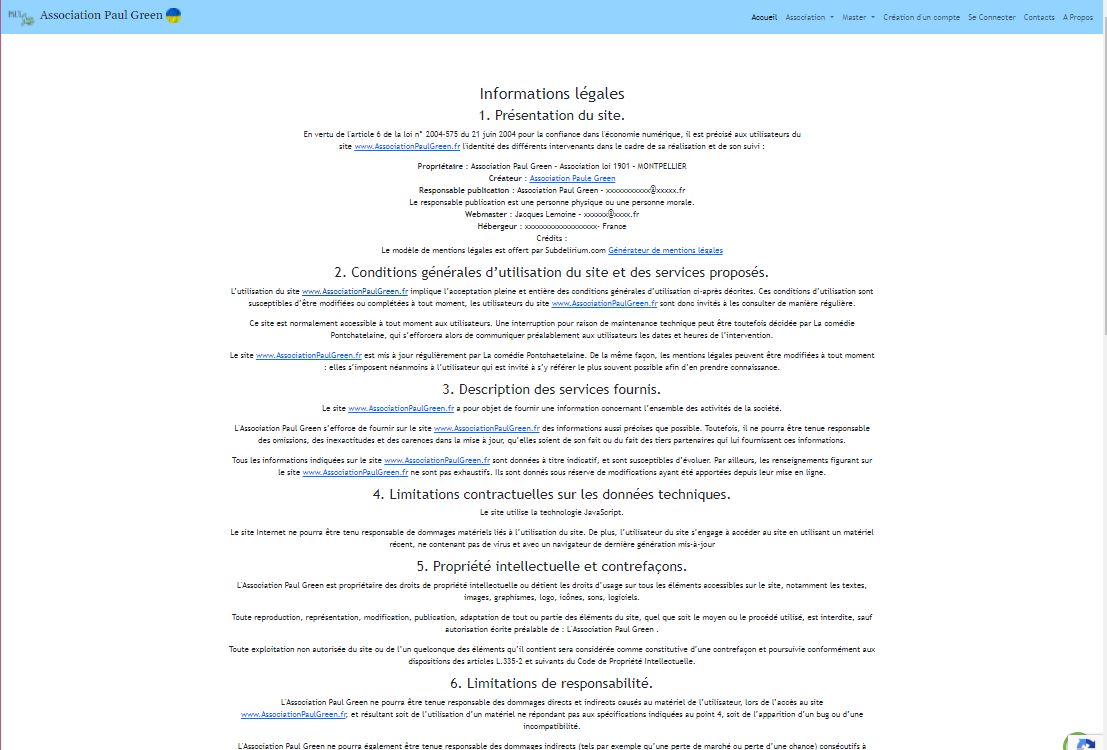
            // =====

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

        });

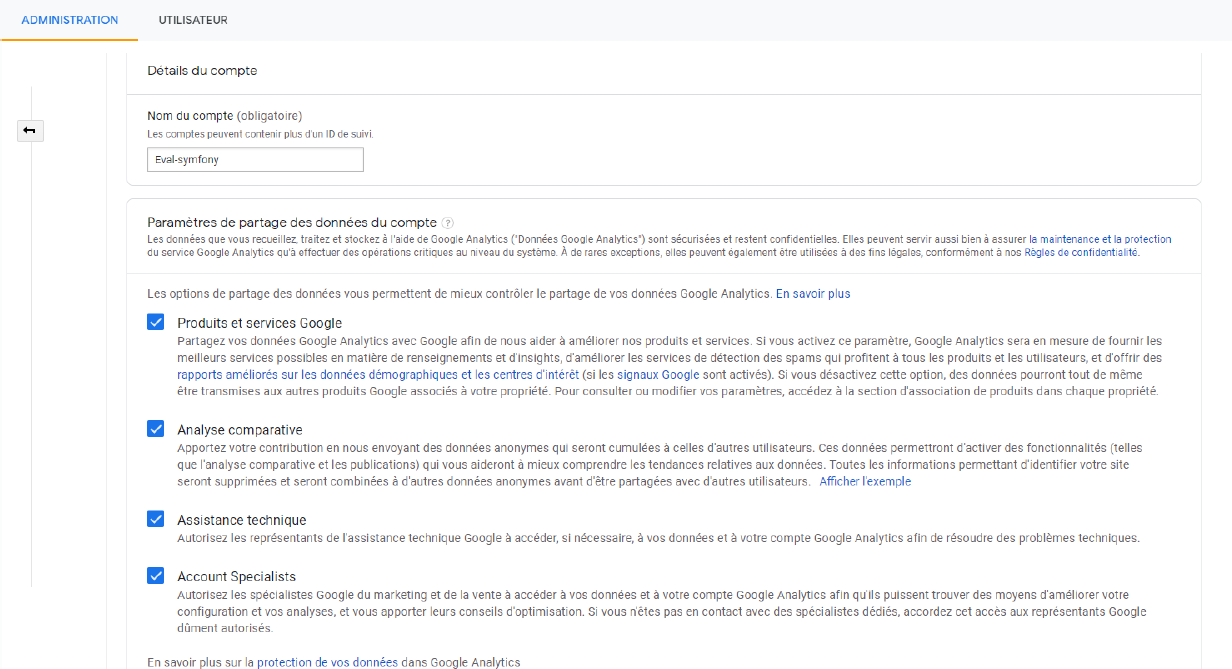
    </script>

##### 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.**

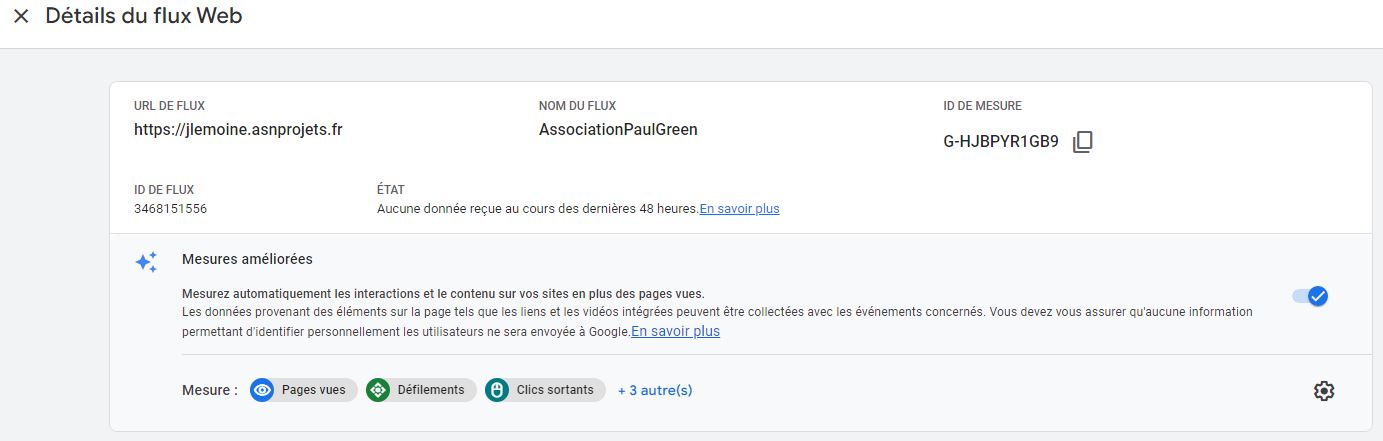


### Intégration de Google Analytics



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>

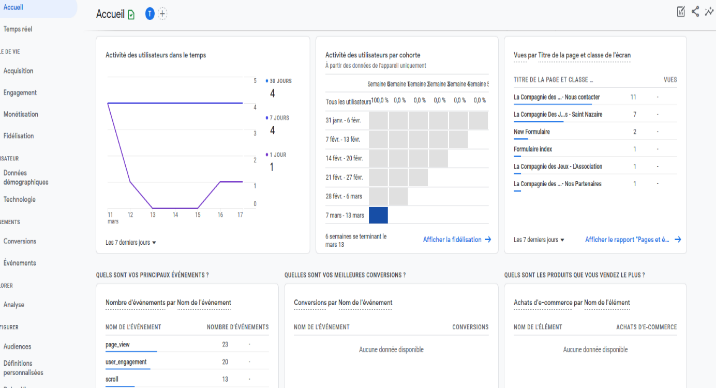
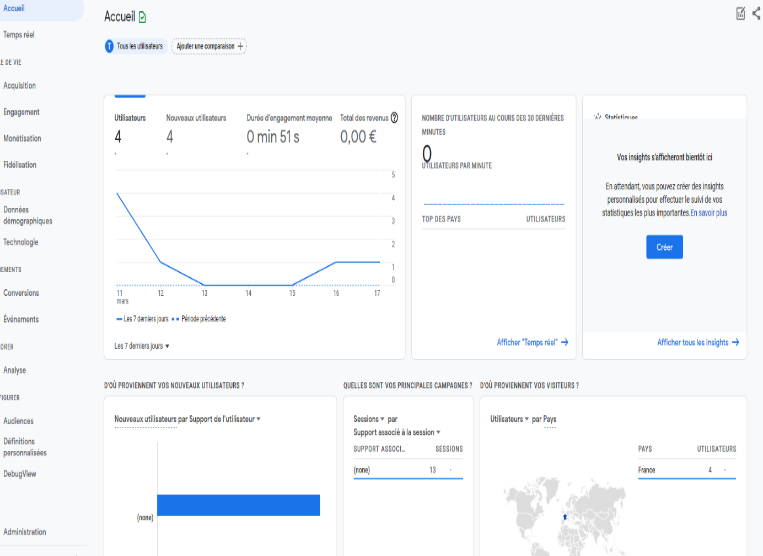
        window.dataLayer = window.dataLayer || [];

        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 sur non significatif.

*aliser une interface utilisateur*

*avec gestion de contenu*

Pour répondre au besoins du site, j’ai mis en place un système de gestion de contenu. L’adhérent ou l’administrateur aura la possibilité d’importer des photos ou logo, de mettre du texte ou de le modifier.

INTERFACE D’ADMINISTRATION

La partie Administration du site est relativement simple.

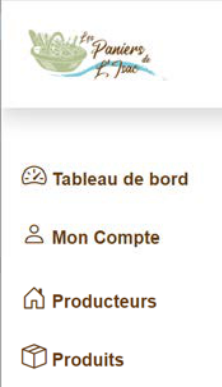
J’ai créé une barre de navigation en deux partie, qui comprend le haut, mais surtout sur la gauche de la page en position fixe.

La partie haute comprend le pseudo du USER, un bouton home qui redirige vers la page d’accueil de la partie «Publique», et un bouton de déconnexion.



La partie gauche de la barre de navigation comprend tout les liens de navigation dans la partie Administration du site, elle est beaucoup plus complète en version Admin, que pour la version Editor ou même user .

**USER** **EDITOR** **ADMIN**



GESTION DES RUBRIQUES

Pour en revenir quelques instants sur la barre de navigation un lien ce présente comme ceci :

Si je clique sur un des liens (Pour une grande partie des liens), j’arrive sur une page d’index ou de modification de la rubrique sélectionnée. Dans mon exemple, je vais prendre les producteurs, et le lien nous amène dans la page d’index des producteurs.



Cette dernière est représenté sous la forme d’un tableau avec des boutons pour interagir avec, pour l’administrateur, il peut voir tous les producteurs, tandis que pour l’Éditeur, il peut voir que les producteur qu’il a créé.

Les boutons pour interagir donne la possibilité ( dans cette exemple ) de Voir la page lié à l’entité (représenté par l’icône « œil », de l’éditer (icône « crayon »), ou bien de la supprimer (icône « poubelle »).



31

Dossier de projet Erwan Gentric - Arinfo - 2021

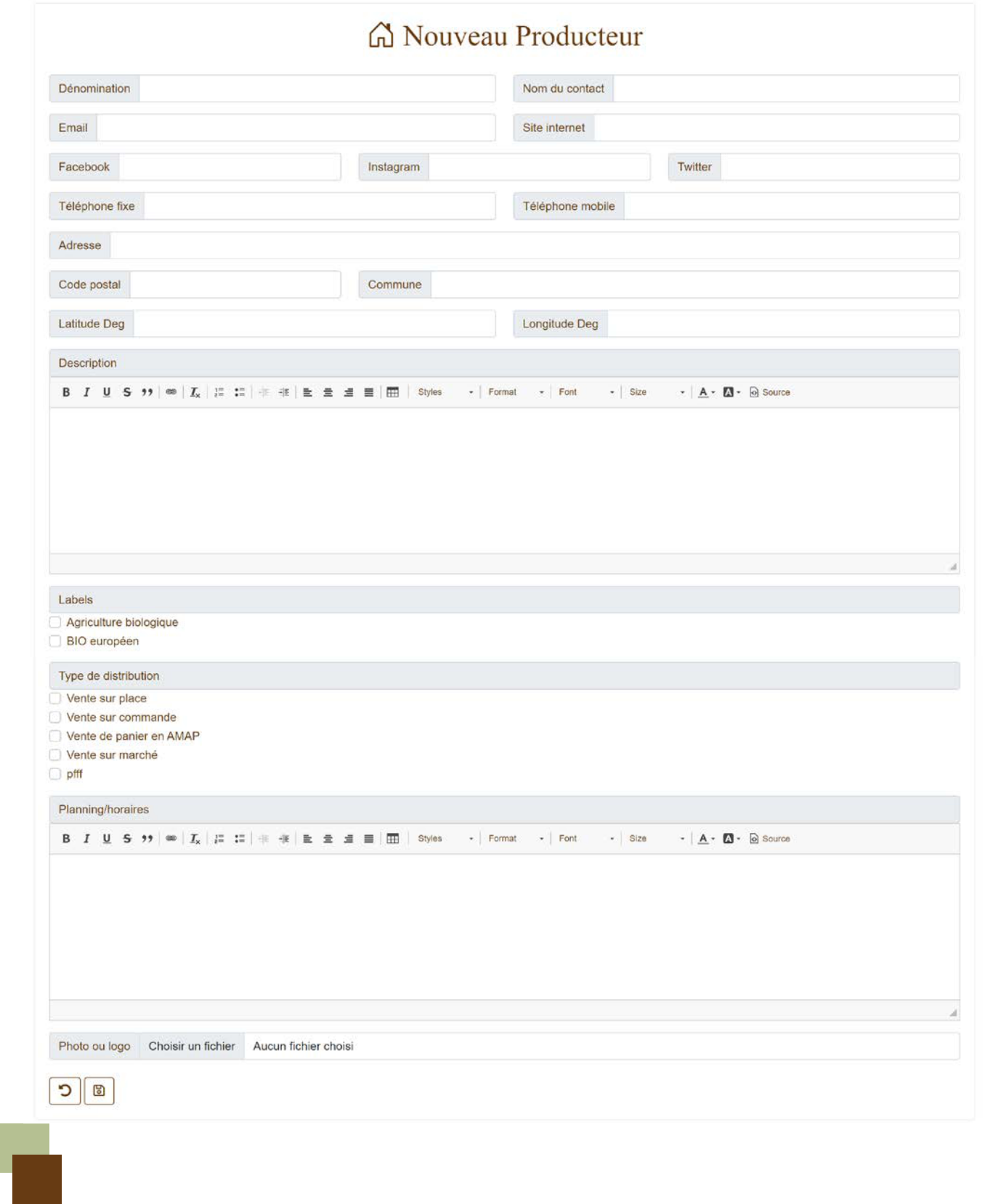
AJOUT D’UNE RUBRIQUE

Afin d’ajouter une entité dans la rubrique, en l’occurrence un producteur pour l’exemple (globalement le concept est le même pour les autres parties de la gestion de contenu), j’ai mis en place un boutons « plus » en bas à gauche du tableau.

En cliquant sur le bouton « + » l’utilisateur est dirigé sur la page d’ajout de contenu, en



l’occurrence « Nouveau Producteur» qui ce présente comme suit :



Ici il y a beaucoup champs à remplir afin d’avoir une fiche détaillée du producteur le premier étant la dénomination du producteur suivi du nom du contact. Après nous avons tout ce qui est lié à internet en commençant par l’e-mail, l’adresse du site internet, l’adresse de facebook, instagram et twitter. Ensuite nous avons les coordonnées téléphonique, fixe et mobile suivi de l’adresse postale.

Il y a aussi deux champs permettant de rentrer les coordonnées GPS du producteur, afin qu’il soit géolocalisé sur une carte.

Il y a le champs description.

On peut aussi sélectionner un label et un type de distribution.

Le champs Planning/horaires.

Il y a aussi la possibilité d’ajouter une photo ou bien le logo du producteur.

Pour les deux champs celui de la description et celui du planning horaires, pour c’est deux champs, j’ai choisi un système de traitement de texte afin d’avoir plus de choix de personnalisation du contenu. J’ai donc utilisé CK Editor, qui est un éditeur de texte open-source, que j’ai installé via composer sous forme de bundle dans symfony :

composer require friendsofsymfony/ckeditor-bundle

Ensuite j’ai vérifié que le bundle était bien affiché à la fin de la liste des bundles dans votre fichiers config/bundles.php

FOS\CKEditorBundle\FOSCKEditorBundle::class => [‘all’ => true],

La deuxième étape est d’installer ckeditor en entrant la commande Symfony suivante :

php bin/console ckeditor:install

Puis on vérifie que les assets soient bien installés avec la ligne de commande suivantes :

php bin/console assets:install public

La troisième étape va consister à configurer ckeditor afin de paramétrer les options auxquels les utilisateurs auront accès , donc dans le fichier config/packages/fos\_ ckeditor.yaml j’ai ajouté le code ci-dessous :

fos\_ck\_editor:

configs:

my\_config:

toolbar:

* { name: «styles», items: [‘Bold’, ‘Italic’, ‘Underline’, ‘Strike’, ‘Blockquote’, ‘- ‘, ‘Link’, ‘-’, ‘RemoveFormat’, ‘-’, ‘NumberedList’, ‘BulletedList’, ‘-’, ‘Outdent’, ‘Indent’, ‘-’, ‘- ‘, ‘JustifyLeft’, ‘JustifyCenter’, ‘JustifyRight’, ‘JustifyBlock’, ‘-’, ‘Image’, ‘Table’, ‘- ‘, ‘Styles’, ‘Format’,’Font’,’FontSize’, ‘-’, ‘TextColor’, ‘BGColor’, ‘Source’] }

J’ai suprimé l’ajout d’image ‘Image’, car je ne voulais pas de cette option.

Après il faut intégrer Ckeditor dans mon formulaire de création de producteur J’ouvre donc mon fichier Form/ProducersType.php et j’ajoute la classe

33

Dossier de projet Erwan Gentric - Arinfo - 2021

CKEditorType à ma propriété descriptionProduct.

Attention de ne pas oublier d’importer les classes afin d’ajouter le « use » nécessaire à l’appel de la classe CKEditorType.

use FOS\CKEditorBundle\Form\Type\CKEditorType;

class ProducersType extends AbstractType

{

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

$builder

->add(‘name’)

->add(‘description’, CKEditorType::class)

Et pour terminer il faut afficher un contenu généré par un ckeditor ce n’est pas compliqué, il suffit d’afficher la donnée comme à notre habitude et d’ajouter la propriété twig « raw » qui permet d’interpréter le langage html.

<tr>

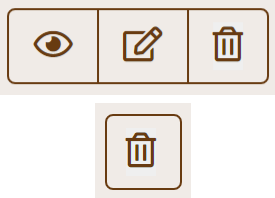
<th>Description</th>

<td>{{ producer.description|raw }}</td>

</tr>

ÉDITION / SUPPRESSION D’UNE RUBRIQUE

Pour éditer ou supprimer une rubrique ou en l’occurrence un producteur, il nous suffit de retourner sur la page d’index des équipes et de cliquer sur le bouton désiré. Il y a deux façon pour supprimer un producteur, soit directement sur la page d’index, soit sur la page d’édition. Je précise que la page d’édition est en tout point similaire à la page d’ajout d’une rubrique, il y a simplement un lien en plus pour supprimer dans la page d’édition.



34

Dossier de projet Erwan Gentric - Arinfo - 2021

*Front -End*

Mise en place du Framework CSS

Symfony ne possède par nativement Bootstrap mais est conçue pour fonctionner très efficacement avec, la mise en place est plutôt simple.

Pour utiliser Bootstrap avec le CDN il me faut poster dans mon <head> cette balises :

<link rel=»stylesheet» href=»https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/ dist/css/bootstrap.min.css» integrity=»sha384-F3w7mX95PdgyTmZZMECAngseQ B83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU» crossorigin=»anonymous»>

Et en bas de page je dois appeller le plugins JavaScript :

<script src=»https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap. bundle.min.js» integrity=»sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJo Mp4YLEuNSfAP+JcXn/tWtIaxVXM» crossorigin=»anonymous»></script>

Nous allons après dans le fichier de configuration de twig qui se trouve dans config/ package/twig.yaml afin d’ajouter cette ligne de code :

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

Mise en place des icons

j’ai également utilisé les icones proposé par Bootstrap et pour cela j’ai du ajouter le CDN dans mon <head>

<link rel=»stylesheet» href=»https://cdn.jsdelivr.net/npm/bootstrap-icons@1.6.1/ font/bootstrap-icons.css»>

J’ai aussi utilisé les icones de Font Awesome et pour cela j’ai également ajouté le code javascript dans mon <head>

<script src=»https://kit.fontawesome.com/cb109f8570.js» crossorigin=»anonymous»></script>

Mise en place des typos

J’ai utilisé les fonts de Google Fonts et pour cela j’ai du ajouter dans mon <head> les codes:

<link href=»https : //fonts.googleapis.com/css2? family= Spartan:wght@100;200;300;400;500;600;700;800;900 & family= Roboto+Slab:wght@100;200;300;400;500;600;700;800;900 & display=swap» rel=»stylesheet»>

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

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

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 sont un plus, mais je ne m’en suis pas servis.

Ma démarche est assez simple je crée un nouveau fichier .scss pour chaque grande partie de mon site et j’ai donc un fichier style.scss pour la partie «Annuaire», et un fichier styleAD.scss pour la partie «Administration» du site.

Sur Visual studio code j’ai installé une extension qui me permet de compiler mon fichier .scss en un fichier .css.

* NAV
* ==========================

nav {

border-bottom: $brown solid 1px;

li {

display: inline-block;

min-width: 50px;

a {

text-decoration: none;

line-height: 30px;

padding: 0 10px;

text-align: center;

font-size: 1rem;

font-weight: 600;

color: $brown !important;

}

}

}

Je trouve pour ma part que le système d’imbrication de Sass particulièrement ingénieux et pratique,

* présent, je l’utilise en permanence.

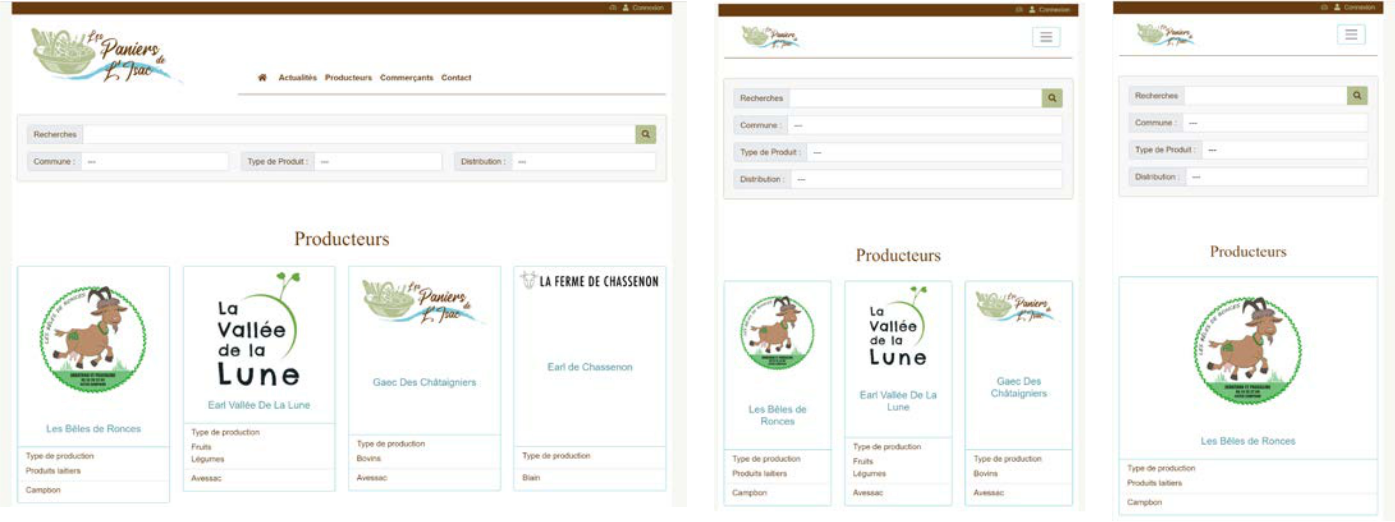
36

Dossier de projet Erwan Gentric - Arinfo - 2021

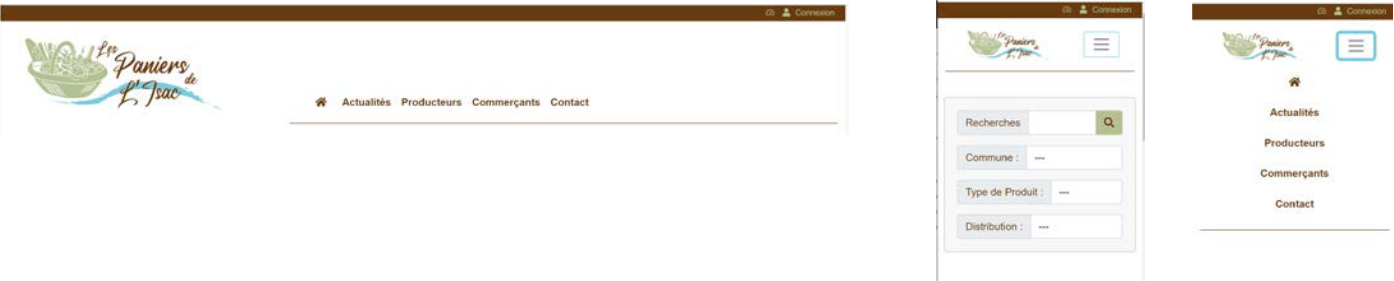
Réaliser une interface web statique et dynamique

Pour que l’expérience utilisateur soit la plus intuitive et interactive possible, il me fallait rendre mon interface responsive adaptable sur plusieurs tailles d’écrans, types d’appareils.

Je me suis servis, tout d’abord, des différentes class Bootstrap «**sm - md - lg - xl**» afin que les éléments s’adapte au mieux à l’écran en fonction du ratio et de l’orientation. Leur utilisation est plutôt simple, on part sur une grille de 12 colonnes qu’il nous faut remplir avec la class « **col** » et des lignes avec la class « **row** », le tout peux être contenu dans une div contenant la class « **container** ». Exemple :



Un autre exemple intéressant de l’utilisation de Bootstrap est la navbar, cette dernière est très facile à mettre en place mais il est plus compliqué de la personnaliser ( idem pour le carrousel ). Le gros avantage de la navbar Bootstrap est qu’elle est responsive et qu’elle affiche donc naturellement un menu burger sur les petits écrans :



Pour l’affichage mobile grace à Bootstrap je n’est presque pas eut à utiliser les media queries, juste pour l’affichage de l’image en background dans mon menu

@media screen and (max-width: 991px) {#slider {

display: none;}

.backImg{

background-image: linear-gradient(to left bottom, rgba(255, 255, 255, 0), rgba(255,255,255,0.7), rgba(255,255,255,0.9), rgba(255,255,255,1), rgba(255,255,255,1)),

url(‘../../images/PhotoHeader.jpg’);

max-height: 100vh;

background-position: center;

background-size: cover;

}

|  |  |
| --- | --- |
| } | 37 |

Dossier de projet Erwan Gentric - Arinfo - 2021



Pour situer la carte dans le cadre il me faut utiliser ce script :

<!-- Fichiers Javascript Open street Map-->

<script src=»https://unpkg.com/leaflet@1.7.1/dist/leaflet.js» integrity=»sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUr Bc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA==» crossorigin=»»>

</script>

<script type=»text/javascript»>

* On initialise la latitude et la longitude (centre de la carte) var lat = {{ producer.longitudeDeg }};

var lon = {{ producer.latitudeDeg }}; var macarte = null;

* Fonction d’initialisation de la carte

function initMap() { // Créer l’objet «macarte» et l’insèrer dans l’élément HTML qui a l’ID «map»

macarte = L.map(‘map’).setView([

lat, lon

], 11);

38

*Exemples de codes*

Recherche FullText

Je voulais pour mon annuaire, faire un champs de recherche multi-mots sur mes rubriques, j’ai cherché sur internet comment le réaliser, et j’ai trouvé un tutoriel de Nouvelle techno et voici comment j’ai procédé.

**Étape 1**

Tout d’abords j’ai été sur mon dossier Entity et sur mon fichier Producers.php,

j’ai ajouter ce code à mon annotation permettant de faire un index plein champs de ma table «producers» et je lui est précisé quel sont les colonnes que je veux indexer (name,city et description) et le type d’indexation (flags=fulltext).

/\*\*

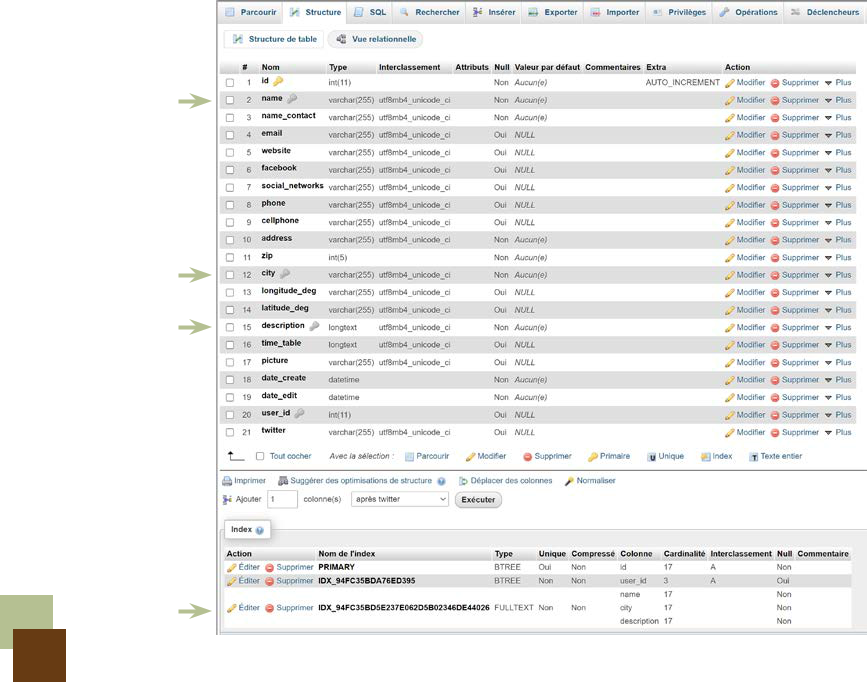
* @ORM\Entity(repositoryClass=ProducersRepository::class)
* @ORM\Table(name=»producers», indexes={@ORM\ Index(columns={«name», «city», «description»}, flags={«fulltext»})})

\*/

Ensuite, je vais dans ma console pour créer un index fulltext sur ma base de données

php bin/console make:migration

php bin/console doctrine:migrations:migrate



**Étape 2**

Il va falloir créer une requête MatchAgainst

Dans le fichier «src», j’ai créé un nouveau dossier «Extensions» puis un dossier «Doctrine» et enfin dans ce dossier j’ai créé un fichier «MatchAgainst.php» dont voici le code

<?php

namespace App\Extensions\Doctrine;

use Doctrine\ORM\Query\AST\Functions\FunctionNode; use Doctrine\ORM\Query\Lexer; use Doctrine\ORM\Query\Parser;

use Doctrine\ORM\Query\SqlWalker;

class MatchAgainst extends FunctionNode {

/\*\* @var array list of \Doctrine\ORM\Query\AST\PathExpression \*/

protected $pathExp = null;

/\*\* @var string \*/

protected $against = null;

/\*\* @var bool \*/

protected $booleanMode = false;

/\*\* @var bool \*/

protected $queryExpansion = false;

public function parse(Parser $parser)

{

// match

$parser->match(Lexer::T\_IDENTIFIER);

$parser->match(Lexer::T\_OPEN\_PARENTHESIS);

* first Path Expression is mandatory $this->pathExp = [];

$this->pathExp[] = $parser->StateFieldPathExpression();

* Subsequent Path Expressions are optional

$lexer = $parser->getLexer();

while ($lexer->isNextToken(Lexer::T\_COMMA)) {

$parser->match(Lexer::T\_COMMA);

$this->pathExp[] = $parser->StateFieldPathExpression();

}

$parser->match(Lexer::T\_CLOSE\_PARENTHESIS); // against

if (strtolower($lexer->lookahead[‘value’]) !== ‘against’) { $parser->syntaxError(‘against’);

}

$parser->match(Lexer::T\_IDENTIFIER);

$parser->match(Lexer::T\_OPEN\_PARENTHESIS); $this->against = $parser->StringPrimary();

if (strtolower($lexer->lookahead[‘value’]) === ‘boolean’) { $parser->match(Lexer::T\_IDENTIFIER); $this->booleanMode = true;

}

if (strtolower($lexer->lookahead[‘value’]) === ‘expand’) { $parser->match(Lexer::T\_IDENTIFIER); $this->queryExpansion = true;

}

$parser->match(Lexer::T\_CLOSE\_PARENTHESIS);

}

public function getSql(SqlWalker $walker)

{

$fields = [];

foreach ($this->pathExp as $pathExp) {

$fields[] = $pathExp->dispatch($walker);

}

$against = $walker->walkStringPrimary($this->against)

. ($this->booleanMode ? ‘ IN BOOLEAN MODE’ : ‘’)

* ($this->queryExpansion ? ‘ WITH QUERY EXPANSION’ : ‘’); return sprintf(‘MATCH (%s) AGAINST (%s)’, implode(‘, ‘,

$fields), $against);

}

}

**Étape 3**

Il va falloir déclarer cette requête dans le fichier doctrine

qui se situe dans config/packages/ doctrine.yaml, et dans l’ORM ajouter une partie «dql»



orm:

auto\_generate\_proxy\_classes: true

naming\_strategy: doctrine.orm.naming\_strategy.underscore\_number\_ aware

auto\_mapping: true

mappings:

App:

 dql:

42

is\_bundle: false

type: annotation

dir: ‘%kernel.project\_dir%/src/Entity’

prefix: ‘App\Entity’

alias: App

string\_functions:

MATCH\_AGAINST: App\Extensions\Doctrine\MatchAgainst

**Étape 4**

Je vais devoir faire un formulaire de recherche, sur mon terminal je tape :

php bin/console make:form

Attention a ne surtout pas l’appeller «SearchType», Moi je l’ai appellé «SearchAllType.php» et préciser qu’il n’intervient sur aucune Entité.

Ensuite j’ai modifié le builder, j’ai ajouté le champ mots, et j’utilise le champs de formulaire «SearchType»

$builder

->add(‘mots’, SearchType::class, [

‘label’ => false,

])

;

Sans oublier de déclarer la class

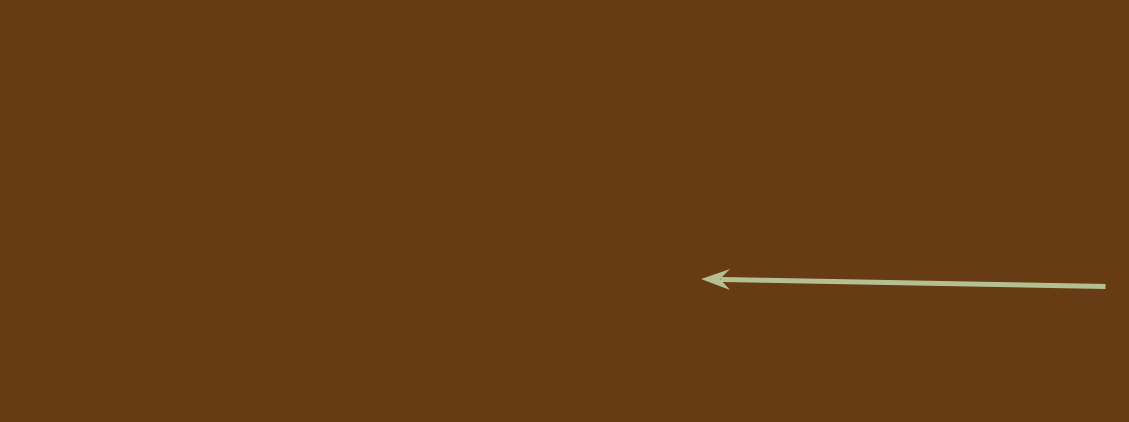
use Symfony\Component\Form\Extension\Core\Type\SearchType;

**Étape 5**

Pour afficher ce formulaire, il va falloir modifier le controller, ici celui de ProducersController.php

|  |  |  |
| --- | --- | --- |
| /\*\* | Attention, penser | |
| \* @Route(«/producersIndex», name=»producersAFF») | à supprimer la | |
| \*/ | methods{«GET»} | |
| public function producersIndex( | dans la Route | |
| Request $request, | Penser à ajouter | |
| TypeDistributionsRepository $typeDistributionsRepository, |
| ProducersRepository $producersRepository, | les dépendances | |
| TypeProductsRepository $typeProductsRepository, |  |  |
| ProductsRepository $productsRepository, |  |  |
| TradersRepository $tradersRepository): Response | variable | |
| { | producers à | |
| $producers = $producersRepository->findAll(); | mettre avant | |
| $form = $this->createForm(SearchAllType::class); | Affichage du | |
|  |
| $search = $form->handleRequest($request); | formulaire | |
|  |  |
| if($form->isSubmitted() && $form->isValid()){ |  |  |
| // on recherche les producteur correspondant au mots | Gestion du | |
| formulaire et | |
| clés | utilisation de la | |
| $producers = $producersRepository->search($search- | méthode search | |
| >get(‘mots’)->getData()); |  |  |
| } |  |  |
|  |  | 43 |



Envoyé à la vue

return $this->render(‘producers/producersIndex.html.twig’, [ ‘producers’ => $producers,

‘products’ => $productsRepository->findAll(), ‘typeProducts’ => $typeProductsRepository->findAll(), ‘typeDistributions’ => $typeDistributionsRepository->findAll(), ‘trader’ => $tradersRepository->findAll(),

‘form’ => $form->createView() ]);

}

Ne pas oublier de déclarer les class

use App\Form\SearchAllType;

use Symfony\Component\HttpFoundation\Request;

Dans mon formulaire, templates/recherch/\_form.html.twig

{{ form\_start(form) }} <div class=»row»>

<div class=»col-12»>

<div class=»input-group»>

<span class=»input-group-text» id=»basic-addon1»>Recherches</span>

{{ form\_widget(form.mots) }}

<button type=»submit» class=»btn mb-0 mt-0»> <i class=»fas fa-search»></i>

</button>

</div>

</div>

</div>

{{ form\_end(form) }}

Dans ma vue, templates/producers/producersIndex.html.twig j’ai fais un «include» de mon formulaire

{{ include(‘recherch/\_form.html.twig’) }}

44

**Étape 5**

Dans le repository je vais créer une Méthode permettant de faire une recherche sur producers (src/Repository/ProducersRepository.php).

/\*\*

* Recherche les producteurs en fonction du formulaire

\* @return void

\*/

public function search($mots){

$query = $this->createQueryBuilder(‘p’); if($mots != null){

$query->where(‘MATCH\_AGAINST(p.name, p.city, p.description) AGAINST

(:mots boolean)>0’)

->setParameter(‘mots’, $mots);

}

return $query->getQuery()->getResult();

}

SELECT2

Pour un formulaire, je voulais utiliser un Select qui me permettait d’écrire les premières lettres d’un producteur et qu’il me l’affiche, et pour un autre Select, je voulais pouvoir choisir plusieurs commerçants, et j’ai trouvé le SELECT2 en javascript qui propose Single select boxes et le Multi-select boxes, voici comment j’ai procédé.

J’ai copié ce code dans mon head:

<link href=»https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min. css» rel=»stylesheet» />

et celui ci en bas de page :

<script src=»https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min. js»></script>

Ensuite j’ai copié ce code Javascript dans mon fichier app.js :

$(document).ready(function () {

* select2 pour les commerçants new product $(‘.select-traders’).select2();
* select2 pour les producteurs new product $(‘.select-producers’).select2();

});

45

Dossier de projet Erwan Gentric - Arinfo - 2021

Ensuite j’ai modifié le builder de mon FormType, sans oublié la Class :

->add(‘producers’, EntityType::class, [

‘class’ => Producers::class,

‘choice\_label’ => ‘name’,

‘query\_builder’ => function(EntityRepository $er){ return $er->createQueryBuilder(‘c’)

->orderBy (‘c.name’, ‘ASC’);

},

‘attr’ => [

‘class’ =>’select-producers’,

],

‘mapped’ => false,

])

->add(‘trader’, EntityType::class, [

‘class’ => Traders::class,

‘multiple’ => true,

‘expanded’ => false,

‘required’ => false,

‘choice\_label’ => ‘name’,

‘query\_builder’ => function(EntityRepository $er){ return $er->createQueryBuilder(‘c’)

->orderBy (‘c.name’, ‘ASC’);

},

‘attr’ => [

‘class’ =>’select-traders’,

]

])

J’ai utilisé QueryBuilder, qui est générateur de requêtes fournit avec Doctrine c’est une manière orientée objet d’écrire des requêtes. Il est recommandé de l’utiliser lorsque les requêtes sont construites dynamiquement (c’est-à-dire basées sur des conditions PHP), ici c’est pour classer la «name» par ordre alphabétique

Filtre de recherche

Filtre de recherche en javascript pour afficher ici les producteurs par commune

Ici, le javascript me permet de masquer mes cards, en remplaçant la class :

$(«#selectCity»).click(function () {

if ($(«#selectCity»).val()!=null) {

$(«.cardlien»).removeClass(«display»);

var idSelect = $(«#selectCity»).val();

var id = «.» + idSelect;

$(id).toggleClass(«display»);

}

});

et dans le fichier twig :

<div class=»col-lg-4»>

<div class=»input-group mt-3»>

<span class=»input-group-text» id=»basic-addon1»>Commune :</span> <select class=»form-control» id=»selectCity»>

<option value=»0» disabled selected hidden>---</option> {% set tableau = [] %}

{% for producer in producers %}

{% if producer.city not in tableau %}

<option value=»{{producer.city}}»>

{{ producer.city }}<br>

{% set tableau = tableau|merge([producer.city])%} {% endif %}

{% endfor %}

</option>

</select>

</div>

</div>

*Securite*

VEILLE SUR LA SÉCURITÉ

Je souhaitais avoir une sécurité optimal pour mon site web, ce qui m’a poussé à faire de la veille en sécurité, on m’a conseillé de prendre en compte les actualités sur le site web OWSAP qui répertorie toutes les failles de sécurités web ainsi que les recommandations pour s’en prémunir.

==> Liste des 10 types de risque liés à la sécurité d’application.

www.owasp.org/www-project-top-ten/

J’ai notamment prit le temps de lire la documentation Symfony, très bien expliqué et très complète ==> www.symfony.com/doc/current/security.html

Il y a aussi un site très intéressant pour les francophones c’est le site asfy.fr (Association Francophone Des Utilisateurs de Symfony ) qui nous donne des petites astuces afin de mieux sécuriser nos sites web et notamment d’autre conseil pour d’autres domaines.

Un autre site qui mérite une petite mention c’est wanadev pour son article très intéressant, sur 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/

TEST UNITAIRE DE SECURITÉ

Il me fallait à présent réaliser des tests un peu plus ciblé sur mon projet, j’ai donc effectué des tests unitaires à l’aide du Framework Open Source PHPUnit. PHPUnit est un framework open source de tests unitaires dédié au langage de programmation PHP. Il va nous permettre de tester certaines fonctionnalités de notre site internet.

Pour l’installer il m’a fallut utiliser le gestionnaire de package Composer, via mon terminal à la racine de mon projet avec ces deux commandes :

composer require —dev symfony/phpunit-bridge

composer require —dev symfony/browser-kit symfony/css-selector

Je vais tester les formulaires, les formulaires sont souvent primordiaux dans le bon fonctionnement des sites. C’est pour cela qu’il est très important de veiller à ce que ceux-ci fonctionnent correctement et qu’ils ne présentent pas de failles de sécurité. Ils faut donc les tester.

Pour cela, dans le dossier « Test », je crée un dossier « Forms » dans lequel j’ai crée un fichier « nomEntitéFormTest ». en réalité le nom importe peu il faut qu’il soit logique en fonction du formulaire que nous voulons tester :

<?php

* tests/Form/ProductsFormTest.php namespace App\Tests\Form;

use App\Entity\Producers;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase; class ProducersFormTest extends KernelTestCase{

public function testNewCategory(){

$producers=(new Producers())

->setName(‘77’)

->setNameContact(‘77’)

->setEmail(‘77’)

->setWebsite(‘77’)

->setFacebook(‘77’)

->setSocialNetworks(‘77’)

->setPhone(‘77’)

->setCellphone(‘77’)

->setAddress(‘77’)

->setZip(‘77’)

->setCity(‘77’)

->setLongitudeDeg(‘77’)

->setLatitudeDeg(‘77’)

->setDescription(‘77’)

->setTimeTable(‘77’)

->setPicture(‘77’);

->setTwitter(‘77’);

49

Dossier de projet Erwan Gentric - Arinfo - 2021

self::bootKernel();

$error = self::$container->get(‘validator’)->validate($producers); $this->assertCount(0,$error);

}

}

Ensuite je lance ma commande :

php bin/phpunit

et j’obtiens :

PS C:\wamp64\www\LesPaniersdelisac> php bin/phpunit

Could not open input file: bin/phpunit

PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

|  |  |
| --- | --- |
| Testing |  |
| . | 1 / 1 (100%) |

Time: 00:04.318, Memory: 54.00 MB

OK (1 test, 1 assertion)

Other deprecation notices (1)

1x: Since symfony/framework-bundle 5.2: Accessing the «validator» service directly from the container is deprecated, use dependency injection instead.

1x in ProducersFormTest::testNewCategory from App\Tests\Form

Tout s’est bien passée en l’occurrence j’ai fait exprès de mettre de bonne information mais j’ai testé auparavant avec des informations erroné et j’ai ensuite ajouté des contraintes de validation.

**Les contraintes de validation**

Pour pouvoir utiliser des contraintes de validation dans mon formulaires, il faut tout d’abord installer un nouveau package à votre projet Symfony grâce à Composer.

composer require symfony/validator doctrine/annotations

Ensuite j’ai été regarder la documentation de Symfony afin de trouver les asserts

correspondant à mes formulaires :

https://symfony.com/doc/current/reference/constraints.html

Ne pas oublier de rajouter la class à l’Entity

use Symfony\Component\Validator\Constraints as Assert;

50

Dossier de projet Erwan Gentric - Arinfo - 2021

*Securite ReCaptcha*



Il me fallait ensuite une solution pour éviter un maximum de robots spam sur mon site, j’ai donc installé un outil nommé ReCAPTCHA. Il s’agit d’un script que l’on appelle sur toutes les pages du site afin de les protéger d’éventuelles attaques.

Installer reCaptcha Google V3 sur son site internet

**Qu’est-ce que reCaptcha ?**

reCAPTCHA protège votre site Web contre la fraude et les abus. Ce service de sécurité utilise un moteur d’analyse des risques avancé et des défis adaptatifs pour empêcher les logiciels malveillants de se livrer à des activités abusives sur votre site Web.

Les utilisateurs légitimes pourront ainsi se connecter, effectuer des achats, afficher des pages ou créer des comptes tandis que les faux utilisateurs seront bloqués.

**Etape 1 : Enregistrer votre site**

Allez sur le site https://www.google.com/recaptcha/about/ en étant connecté

* son compte Gmail sur chrome puis allez dans la console d’administration (lien « Admin Console »).

Remplir le formulaire et cliquez sur « Envoyer ».

**Etape 2 : Intégration du reCaptcha sur votre site**

Copiez la clé du site que j’ai généré en remplissant le formulaire d’inscription.

Je copie le script dans le head de mon site.

<script src=˝https://www.google.com/recaptcha/api.js?render=CLE DU SITE ̋ > </script>

Puis je copie ce bout de code dans mon fichier

<script>

function onClick(e) {

e.preventDefault();

grecaptcha.ready(function() {

grecaptcha.execute(‘CLE DU SITE’, {action: ‘submit’}).

then(function(token) {

console.log(‘captacha’);

});

});

}

</script>

*Mise en Production*

Pour ce qui est de la mise en production j’ai longuement hésité entre OVH et o2switch, mon choix s’est finalement porté sur 02switch, son offre me paraissait plus intéressante et plus complète qu’OVH.



O2switch est une entreprise 100% Française autant pour le serveur que pour le SAV.

Une phase importante du développement est le recettage, c’est ce qui permet d’engager une multitude de test que ce soit au niveau du sport (Mobile, Tablette) mais aussi faire des tests de sécurité, tout ceci permettant de vérifier si son site ne comporte pas de failles ( j’en parle un peu plus dans la partie sécurité ).

Pour cette mise en production il me fallait un logiciel qui me permette de me connecter au serveur FTP de l’hébergement o2switch, pour ce faire j’ai utilisé le

logiciel WinSCP qui est un logiciel client FTP et SFTP pour la plateforme Windows.



Sur le mac j’ai utilisé FileZilla.Le client FileZilla prend non seulement en charge FTP, mais également FTP sur TLS (FTPS) et SFTP. Il s’agit d’un logiciel open source distribué gratuitement.



Je m’en suis servis pour envoyer l’intégralité de mon projet Symfony, dans le dossier de mon sous-domaine lespaniersdelisac.erwangentric.fr, mis à part le dossier var et vendor que je trouvais trop volumineux et que je pouvais rétablir via la ligne de commande « composer update » via le terminal ( disponible sur l’interface Cpanel ) mis à disposition par o2switch une fois mes fichiers dans le serveur.

Il me fallait ensuite créer un fichier .htaccess et le placer dans le dossier « public

* depuis la racine de mon dossier Symfony. Le fichier .htaccess est un fichier de configuration (Apache) permettant de définir des règles bien spécifiques dans un répertoire. Ce type de fichier peut être utilisé pour réaliser des redirections ou protéger un répertoire par un mot de passe.

Il me fallait également un fichier .env à la racine de mon projet afin de configurer l’accès à la base de donnée.

J’ai fait attention à l’utilisation des balises « h » pour les titres, un seul h1 et un ordre logique pour les suivantes.

J’ai ajouté aussi des descriptions dans chaque image pour qu’elles soient indexées par le moteur de recherche par le biais d’une balise « alt ».

J’ai mis en place la meta viewport pour l’accessibilité au mobile.

J’ai installé une balise <title> dans chaque page du site.

Tout les liens ont des textes descriptifs et sont « crawlable » .

J’ai précisé la langue du site via la balise <html lang=‘’fr‘’ >.

J’ai ajouté un URL canonique, plutôt important pour le référencement.

Et j’ai ajouté une favicon pour la plupart des navigateurs et une favicon pour apple.

*Test de Performance*

Pour vérifier la rapidité d’affichage, l’accessibilité, les bonnes pratiques au niveau du code ainsi que le référencement, j’ai utilisé le module Lighthouse cité un peu plus haut, pour noter un évolution vis à vis de la performance, j’ai réalisé un premier test de performance :

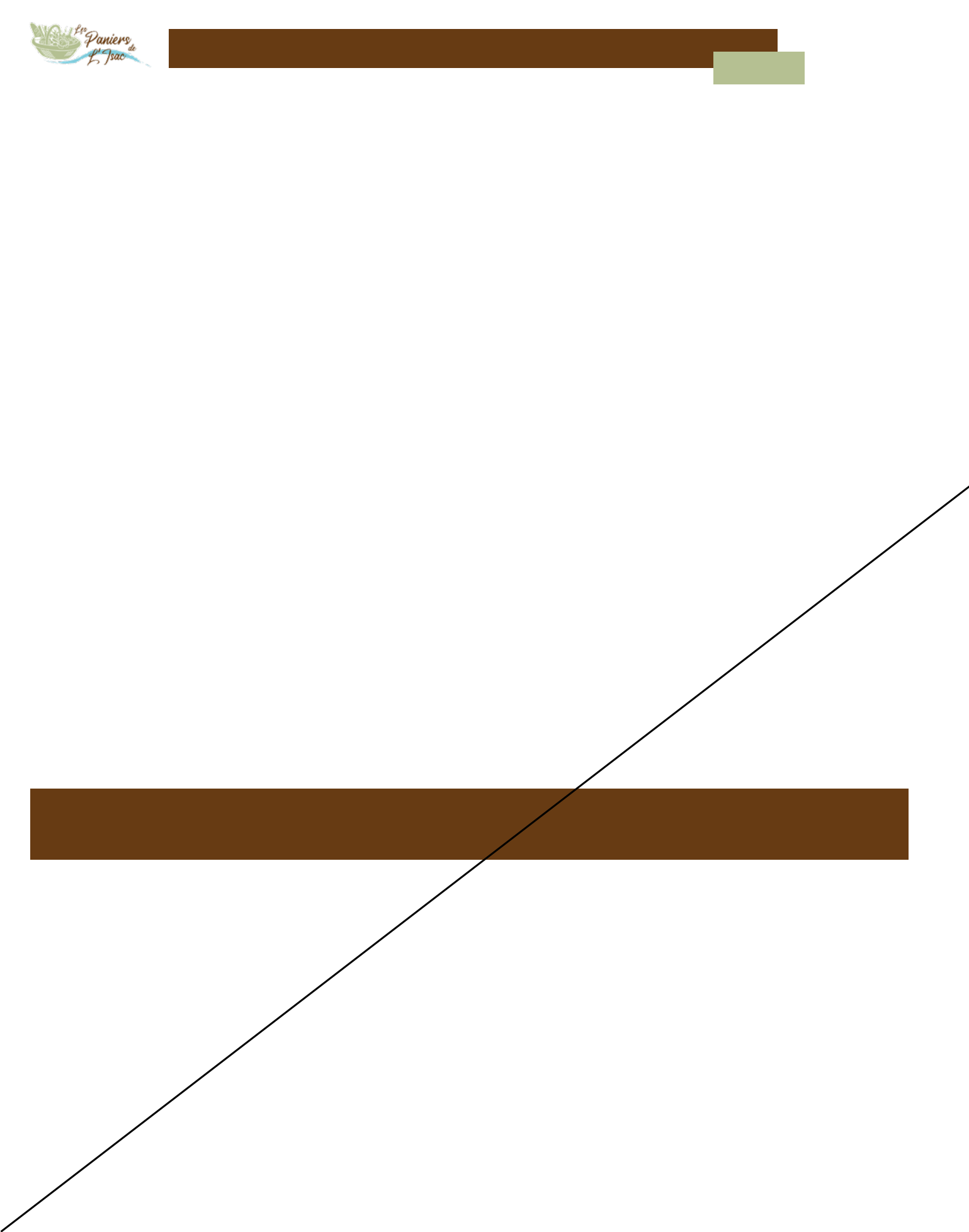
Les performances n’étant clairement pas convaincante, en tout cas pour la version mobile, j’ai décidé de suivre les recommandations de Google afin d’augmenter mon score de performance ainsi que de l’accessibilité.

Pour commencer j’ai redimensionner les images puis je les ai compresser avec photoshop.

Il me fallait ensuite régler le problème des images « hors écran » ainsi que des iframes qui prenaient beaucoup de ressources lors du chargement de la page. J’ai découvert qu’il était possible de différer leurs chargements mais j’ai du faire quelques recherches pour ça que je décrirais dans la partie de recherche en Anglais.

J’ai découvert qu’il était possible d’ajouter un simple attribut dans les balise <img> et <iframe> il s’agit du l’attribut « loading » avec la valeur « Lazy ». Cet attribut est depuis peu nativement pris en charge par les navigateurs Chromium ainsi que Firefox, Safari est en cours de déploiement.

L’attribut loading avec la valeur lazy pour différé le chargement de l’iframe et de l’image.

*Recherche en Anglais*

Dans un désir de vouloir optimiser ma page web afin d’améliorer mon score de Performance sur Lighthouse, il me fallait réduire mon temps de chargement, c’est ainsi qu’après quelques recherches j’ai découvert l’attribut mentionné plus haut « loading ».

La plupart des ressources trouvé n’était pas très précise et il me fallait des informations de meilleurs qualités ainsi que la manière de le mettre en place. C’est ainsi que je suis tombé sur la page « Lazy loading » de Mozilla, cette page est très intéressante et contenait les informations que je désirais avec une traduction de ma part au préalable, car malgré que ce soit une documentation Mozilla, celle-ci n’est pas disponible en Français pour le moment.

La traduction m’a permis de mettre en place le chargement différé sur les images hors écran ainsi que des iframes :

Je suis tombé sur ce paragraphe :

**Images and iframes**

Very often, webpages contain many images that contribute to data-usage and how fast a page can load. Most of those images are off-screen (non-critical), requiring user interaction (an example being scroll) in order to view them.

**Loading attribute**

The loading attribute on an <img> element (or the loading attribute on an <iframe>) can be used to instruct the browser to defer loading of images/iframes that are off-screen until the user scrolls near them.

<img src=»image.jpg» alt=»...» loading=»lazy»>

<iframe src=»video-player.html» title=»...» loading=« lazy»></iframe>

The load event fires when the eagerly-loaded content has all been loaded; at that time, it’s entirely possible (or even likely) that there may be lazily-loaded images that are within the visual viewport that haven’t yet loaded.

Que j’ai traduis par :

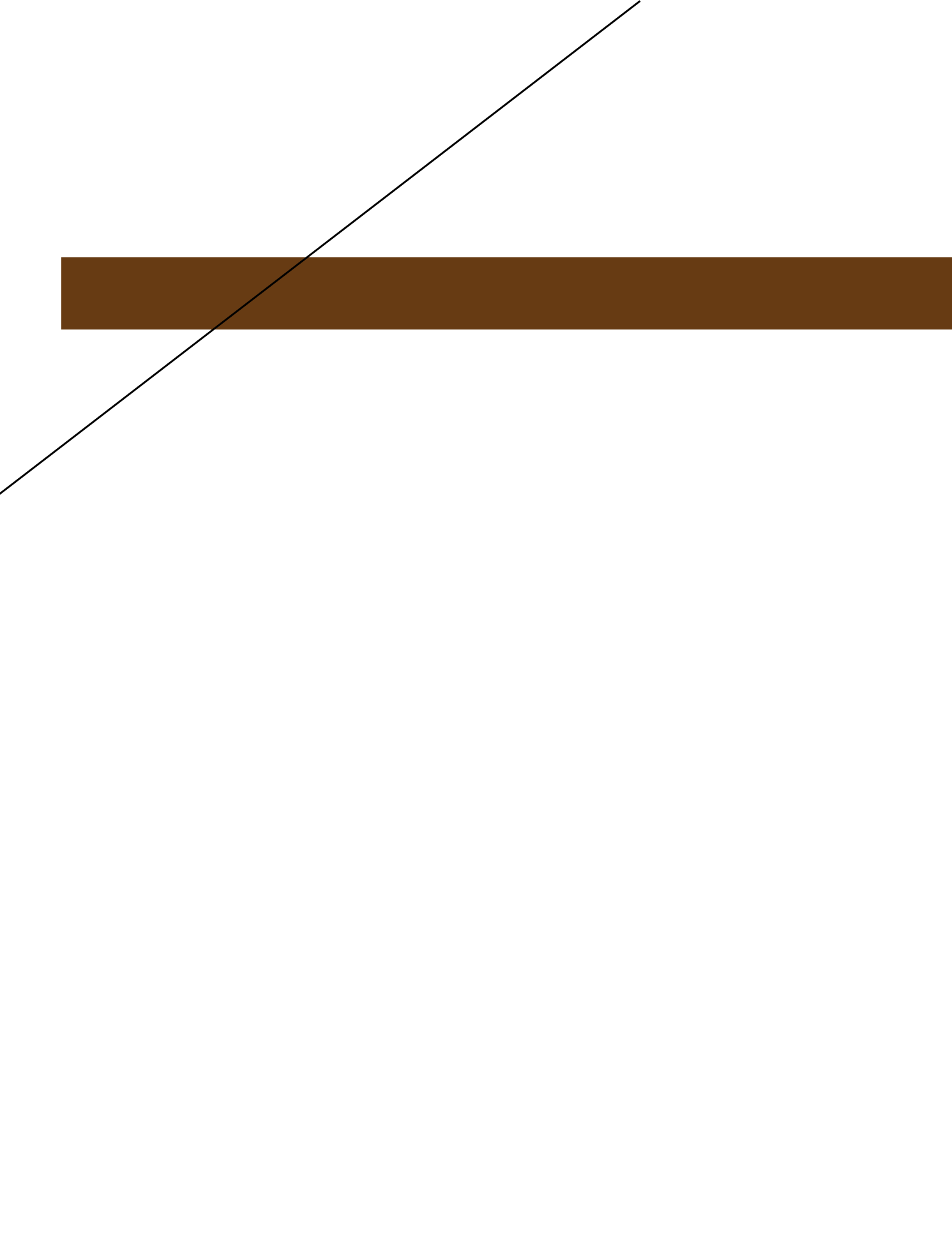
**Images et Iframes**

Très souvent, les pages Web contiennent de nombreuses images qui contribuent

* l’utilisation des données et à la vitesse de chargement d’une page. La plupart de ces images sont hors écran ( non-critique ), nécessitant une interaction de l’utilisateur (un exemple étant le scroll ou défilement ) afin de les voir ou de les visualisés.

56

Dossier de projet Erwan Gentric - Arinfo - 2021



**Attribut de chargement**

L’attribut loading sur l’élément <img> (ou l’attribut loading sur un <iframe>) peut être utilisé pour demander au navigateur de différer le chargement des images / iframes qui sont hors écran jusqu’à ce que l’utilisateur les fasse défiler.

<img src=»image.jpg» alt=»...» loading=»lazy»>

<iframe src=»video-player.html» title=»...» loading=« lazy»></iframe>

L’événement de chargement se déclenche lorsque le contenu chargé avec « empressement » à été chargé; à ce moment-là, il est tout à fait possible (ou même probable) qu’il y ait des images chargées « paresseusement » qui se trouvent dans l’écran visible qui ne sont pas encore chargées.

*Conclusion*

Ce rapport m’a permis d’exposer tout les étapes de créations de mon projet pour «les paniers de l’Isac. Ce projet m’a permis de mettre en avant les producteurs locaux ainsi que les commerçants.

Les trois grandes étapes de mon projet consistaient en premier lieu d’établir le cahier des charges.

Dans un second temps, connaître et déterminer les besoins fonctionnels du site. Pour ce faire j’ai réalisé des diagrammes me permettant d’élaborer l’interface utilisateur et d’établir les différents rôles.

Pour finir, la dernière étape était la conception du site, en prenant en compte les aspects de sécurité, d’ergonomie, de responsive et de test.

Mon projet a été une expérience très enrichissante malgré, les difficultés et les contraintes de temps, je garde une impression très positive de ce projet.

Travailler sur un projet de bout en bout et de maîtriser chaque étape de production m’a permis de rapidement progresser et également de voir beaucoup plus clair vis à vis d’une conception similaire dans le monde professionnel,

Le principal défi était à la fois d’apprendre à coder sous Symfony et de respecter les délais du projet.

Pour la suite, je ne compte absolument pas m’arrêter là, je suis déterminé à améliorer mes compétences en Php et Javascript !

Ce sont des langages qui m’intéresse, et lors de mon stage je me suis aperçu que j’avais encore beaucoup de chose à apprendre, pour le javascript c’est un langage que je n’ai pas pu assez mettre en pratique durant ma formation, et que j’aimerais maîtriser.