**Formation DWWM 2020/2021**

**VANDERMOETEN KERCKHOVE Jérémy**



# SOMMAIRE

[SOMMAIRE 2](#_Toc61430148)

[INTRODUCTION 4](#_Toc61430149)

[1-PRESENTATION 5](#_Toc61430150)

[1.1- Présentation personnelle 5](#_Toc61430151)

[1.2- Présentation de la formation 5](#_Toc61430152)

[1.2 – Mode 83 5](#_Toc61430153)

[1.3- Grande école du numérique 6](#_Toc61430154)

[1.4 Compétences à démontrer pour cette formation 6](#_Toc61430155)

[1.5 Présentation de l’entreprise 7](#_Toc61430156)

[1.6 Description du besoin lors de ce stage 7](#_Toc61430157)

[2. Implémentation d’un module de e-commerce 8](#_Toc61430158)

[2.1 Connexion à l’interface Wordpress du client 8](#_Toc61430159)

[2.2 Installation du plugin Woocommerce 9](#_Toc61430160)

[2.2.1 Création des pages e-commerce 9](#_Toc61430161)

[2.3 Ajout de produits 10](#_Toc61430162)

[2.4 Ajout d’une catégorie de produit 12](#_Toc61430163)

[2.5 Ajout de nouvelles caractéristiques sur le produit 12](#_Toc61430164)

[2.5.1 Ajout par le code 12](#_Toc61430165)

[2.5.2 Ajout de champs au travers d’un plugin 14](#_Toc61430166)

[2.6 Affichage des informations du produit dans la description 14](#_Toc61430167)

[2.6.1 Accéder aux fichiers WooCommerce 14](#_Toc61430168)

[2.7 Configuration des méthodes de livraison 17](#_Toc61430169)

[2.7.1 Test de transporteurs au sein de différentes commandes 18](#_Toc61430170)

[2.8 Installation d’un module de paiement 19](#_Toc61430171)

[3 Modifications au sein des Templates WooCommerce 20](#_Toc61430172)

[3.1 Modification de la Titlebar 20](#_Toc61430173)

[3.2 Modification de la page « Mon compte » 21](#_Toc61430174)

[3.3 Utilisation des médias Queries 22](#_Toc61430175)

[4 Utilisation de Javascript 22](#_Toc61430176)

[4.1 Extrait d’un site Anglophone dans le cadre d’une recherche 24](#_Toc61430177)

[Extrait anglophone 24](#_Toc61430178)

[Traduction de l’extrait 24](#_Toc61430179)

[5 Ajout de données utilisateurs lors d’une commande 26](#_Toc61430180)

[6 Utilisation d’un Framework PHP 27](#_Toc61430181)

[6.1 L’architecture MVC 27](#_Toc61430182)

[6.2 Codage de la maquette 28](#_Toc61430183)

[7 Utilisation des diagrammes UML 29](#_Toc61430184)

[7.1 Le diagramme de cas d’utilisation 29](#_Toc61430185)

[7.2 Le diagramme de séquence système 30](#_Toc61430187)

[7.3 Le diagramme de classe participantes 30](#_Toc61430188)

[8 Livewire 31](#_Toc61430189)

[9 Ajout d’un système d’authentification 33](#_Toc61430190)

[7 Utilisation de l’outil GIT 33](#_Toc61430191)

[Conclusion 34](#_Toc61430192)

[6 Remerciements 34](#_Toc61430193)

# 

# INTRODUCTION

Nous vivons dans une ère incroyable, ou des humains peuvent communiquer en étant à l’autre bout du monde. Cette véritable révolution engendre sans nul doute un besoin, le besoin d’être visible sur internet. Pour cela, il faut faire appel à des gens qui disposent de connaissances bien précises. Ces dernières nous permettent de créer, avec pour seule limite l’imagination du créateur (ainsi que la puissance de sa machine). Ces connaissances permettent de ne pas être seulement spectateur de la révolution technologique qui est en train de se produire, mais aussi d’en être l’acteur.

Dans le cadre de cette formation, un stage en entreprise d’une durée de six semaines a été effectué. Ce dernier s’est déroulé dans l’entreprise « Simon Web Agency ».

Ce dossier expose les compétences que j’ai acquises lors de ma formation, afin d’obtenir le diplôme de Développeur Web, Web mobile et e-tourisme. La plupart des fonctionnalités que j’ai réalisées ont été faites lors de mon stage en entreprise.

Ce stage s’est exclusivement déroulé sur le CMS WORDPRESS, me permettant d’aborder beaucoup de spécificités liées à la création de boutique en ligne.

J’ai donc implémenté un module de e-commerce, dans un site déjà existant utilisant la technologie Wordpress. Ainsi, j’ai mis en œuvre l’installation du module, la création des différentes pages d’e-commerce, la création de produits qu’ils soient simples ou variables, la modification de configuration de produit en ajoutant des champs permettant l’affichage de certaines données sur des produits bien spécifiques, la configuration des méthodes de livraison prenant en compte plusieurs transporteurs, l’un étant agréé pour prendre en compte certains types de produits.

Des modifications ont aussi été effectuées directement dans les fichiers des plugins utilisés, ou encore dans le fichier de configuration de WORDPRESS, ainsi que dans l’espace utilisateur du site.

Enfin, je vais démontrer d’autres compétences en utilisant le Framework PHP Laravel, notamment en réalisant une maquette faite antérieurement, et en utilisant les diagrammes UML afin d’implémenter un système de recherche à mon site. Pour réaliser ce dernier, j’ai utilisé un Framework de LARAVEL, qui se nomme LIVEWIRE. Un espace membre a aussi été réalisé dans ce site, avec le Framework FORTIFY. Je vous montrerai aussi l’utilisation de l’outil de versionning GIT, utilisé lors de ce projet.

# 1-PRESENTATION

## 1.1- Présentation personnelle

Je m’appelle Jérémy VANDERMOETEN KERCKHOVE. Ancien pâtissier, les contraintes de ce métier m’ont donné envie de changer de voie professionnelle. Pendant ma réflexion, je me suis intéressé par simple curiosité au monde du développement Web.

Ayant tout de suite accroché et compris que c’était un métier d’avenir, j’ai décidé d’intégrer une formation professionnelle pour en faire mon métier.

## 1.2- Présentation de la formation

La formation Développeur Web, Web Mobile & e-tourisme est d’une durée de 10 mois, de mai 2020 à février 2021. Elle est composée de 12 stagiaires ayant tous un niveau de départ différent dans le développement web. Au vu des conditions sanitaires difficiles, la majorité de la formation s’est déroulée en télétravail.

Différentes technologies sont étudiées au cours de cette formation, à savoir les langages HTML, CSS, JAVASCRIPT, PHP ou encore MY SQL.

## 1.2 – Mode 83

Le Centre de formation MODE est un organisme de formation professionnelle depuis 2000.

Il propose un large choix de formations numériques, et c’est avec cet organisme que se déroule la formation DWWM 2020/2021, en partenariat avec la GRANDE ECOLE DU NUMERIQUE.

Créée en 1996, l'association MODE 83 (Méditerranée Ordinateurs pour le Développement et l’Emploi) implantée à Draguignan, a pour but de promouvoir et de développer les usages numériques dans le Département du Var. Depuis plus de 20 ans, en tant qu’acteur de développement local, MODE accompagne quotidiennement les publics, les entreprises, les associations et les collectivités de son territoire dans leurs littératies numériques.

Accompagnée depuis 2002 par la Région PACA, grâce aux dispositifs ERIC (Espaces Régionaux Internet Citoyen, depuis 2015, SudLABS) elle anime et gère 4 lieux de médiation numérique. Depuis 16 ans, MODE 83 est un opérateur du plan régional de formation. Son Centre de Formation Numérique délivre sept formations labellisées « Grande École Numérique » à Saint Raphaël, Toulon et Draguignan.

En 2014, MODE 83, créé le 1er FabLab du Var, où les techniques numériques innovantes peuvent être démocratisées et accessibles, il est proposé aux publics de découvrir l’impression 3D, la modélisation et la programmation.

Depuis 2015, MODE a créé et anime un tiers-lieu dédiée aux coopérations numériques. Les trois secteurs de la structure sont :

1. Un Centre de Formation dédié aux métiers du numérique

2. La médiation numérique auprès des publics

3. Le Conseil & Développement Numérique

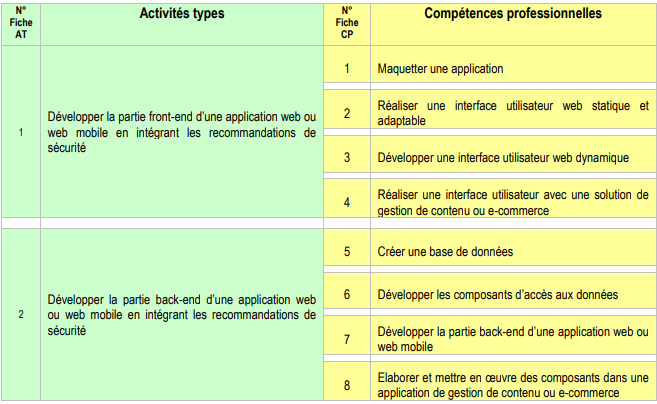
## 1.3- Grande école du numérique



C’est un organisme qui permet à la fois de financer des formations, mais aussi de déposer un label concernant ces dernières. Ce groupe a été mis en place par le gouvernement français en 2015. L’objectif étant de de proposer des méthodes pédagogiques innovantes, prioritairement à des personnes éloignées de l’emploi.

## 1.4 Compétences à démontrer pour cette formation

Pour acquérir le diplôme à l’issue de cette formation, un nombre bien précis de compétences sont à démontrer tout au long d’un rapport, qui sont :



## 1.5 Présentation de l’entreprise



Simon Web Agency est une micro-entreprise située à Cannes, et ne comprend donc qu’un gérant, Mr Simon METIVIER.

Ce dernier travaille essentiellement avec le CMS\* WORDPRESS\*, sur des sites e-commerces visant à vendre des produits en ligne ou encore sur quelques sites dits « Vitrines » de présentation d’entreprise ou d’associations.

Cette entreprise fonctionne la plupart du temps à distance, ses clients n’étant que très rarement proches de son bureau.

# 1.6 Description du besoin lors de ce stage

Lors de ce stage, j’ai pu travailler sur des sites des clients de cette entreprise. Le but premier était donc d’implémenter un module d’e-commerce pour une entreprise possédant déjà plusieurs boutiques dans le sud de la France, et vendant des produits pour le jardinage ou encore d’entretien de piscine.

Cette entreprise vendant des produits dit « chimiques » (Sceau de chlore, produits d’entretien de piscine etc..), différents transporteurs devront donc être utilisés selon le type de produits vendus aux clients, le poids final du colis devra aussi influencer le choix du transporteur. Le premier de prenant pas en compte les colis de plus de trente kilos.

Des catégories de produit devront aussi être créées afin de pouvoir trier ces derniers dans la boutique en ligne, ainsi que des pages correspondantes aux différentes marques de produits vendus sur le site.

Diverses tâches sur plusieurs sites de clients m’ont également été demandées lors de ce stage, principalement sur des ajustements graphiques, me demandant d’utiliser le langage « CSS » afin de notamment adapter des pages sur plusieurs formats d’écrans.

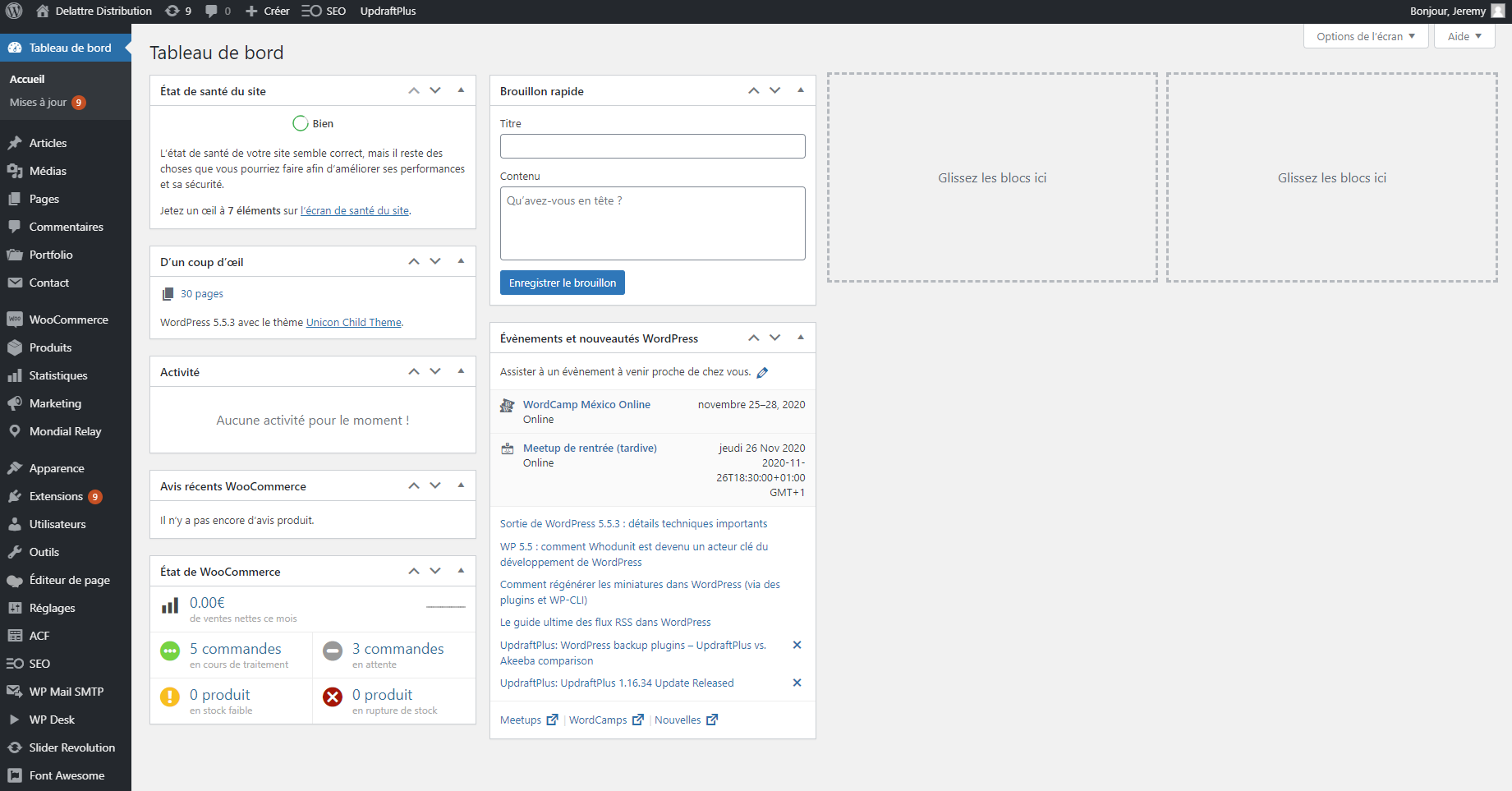
# 2. Implémentation d’un module d’e-commerce

## 2.1 Connexion à l’interface Wordpress du client

Le site étant créé avec le CMS Wordpress, j’ai d’abord dû me connecter à l’interface administrateur. Pour cela, mon tuteur de stage m’a créé des identifiants que j’ai pu rentrer sur la page du site : [www.pagedusite.com/wp-admin](http://www.pagedusite.com/wp-admin).



Une fois ces identifiants rentrés, j’ai eu accès à l’interface me permettant d’effectuer toutes les modifications dans le site.



Interface administrateur Wordpress.

## 

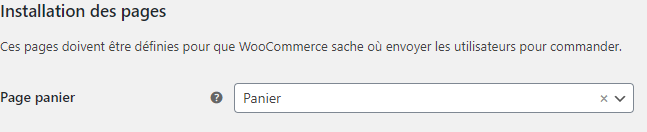
## 2.2 Installation du plugin WooCommerce

J’ai utilisé le plugin « Woocommerce », Pour cela, j’ai dû trouver le plugin dans les extensions de Wordpress. Une fois le module installé, j’ai dans un premier temps pu rentrer toutes les informations du client au sein ce dernier.

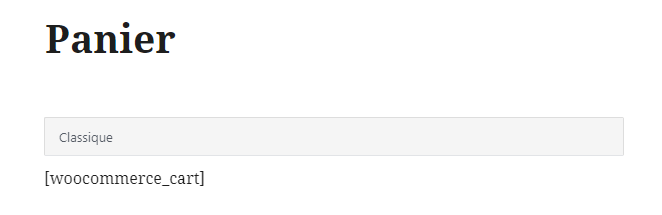
## 2.2.1 Création des pages e-commerce

Le module WORDPRESS met à disposition des « shortcodes\* » permettant de générer les pages correspondantes au e-commerce. Ces codes sont disponibles dans les réglages de WooCommerce.

Pour créer une page, il suffit d’aller dans l’onglet « Pages », et de cliquer sur « Ajouter une page ». Après avoir généré une page « Panier », j’ai pu relier cette dernière avec le module Woocommerce directement dans les réglages, afin que le plugin sache où rediriger les utilisateurs.



Il m’a ensuite fallu rentrer le code correspondant fourni par WooCommerce sur la page en question.



Ainsi, la page Panier est générée. Implémentant toutes les fonctionnalités relatives, comme la gestion du nombre de produit, ou encore la possibilité de calculer les frais de livraison. La même procédure a donc été reproduite pour toutes les autres pages en relation avec le module d’ecommerce, à savoir la page contenant tous les produits et la page de validation de commande.

## 2.3 Ajout de produits

Afin d’ajouter des produits dans la boutique, j’ai dû me rendre dans la section « Produits ».

En cliquant sur « Ajouter un produit », j’accède à une page me demandant un certain nombre d’informations. C’est ici que j’ai pu rentrer toutes les données des produits.

Plusieurs types de produits peuvent être rentrés, j’en ai personnellement utilisé deux types :

**Les produits simples qui** sont utilisés lorsque le produit est unique, et qu’il ne comprend pas de variations.

**Les produits variables**, utilisés lorsque le produit est vendu sous plusieurs formes (Ici des sceaux de chlore vendus sous plusieurs conditionnements par exemple). Cela nous permet de ne générer qu’une seule page d’un seul produit permettant à l’utilisateur de ne pas se perdre dans une page contenant des tas de produits similaires.

Afin de créer des variations de produit, j’ai d’abord dû créer de nouveaux attributs, qui seront utilisés lors des variations de produits. Pour créer un attribut, j’ai d’abord dû leur donner un nom, pour ensuite configurer différents termes.

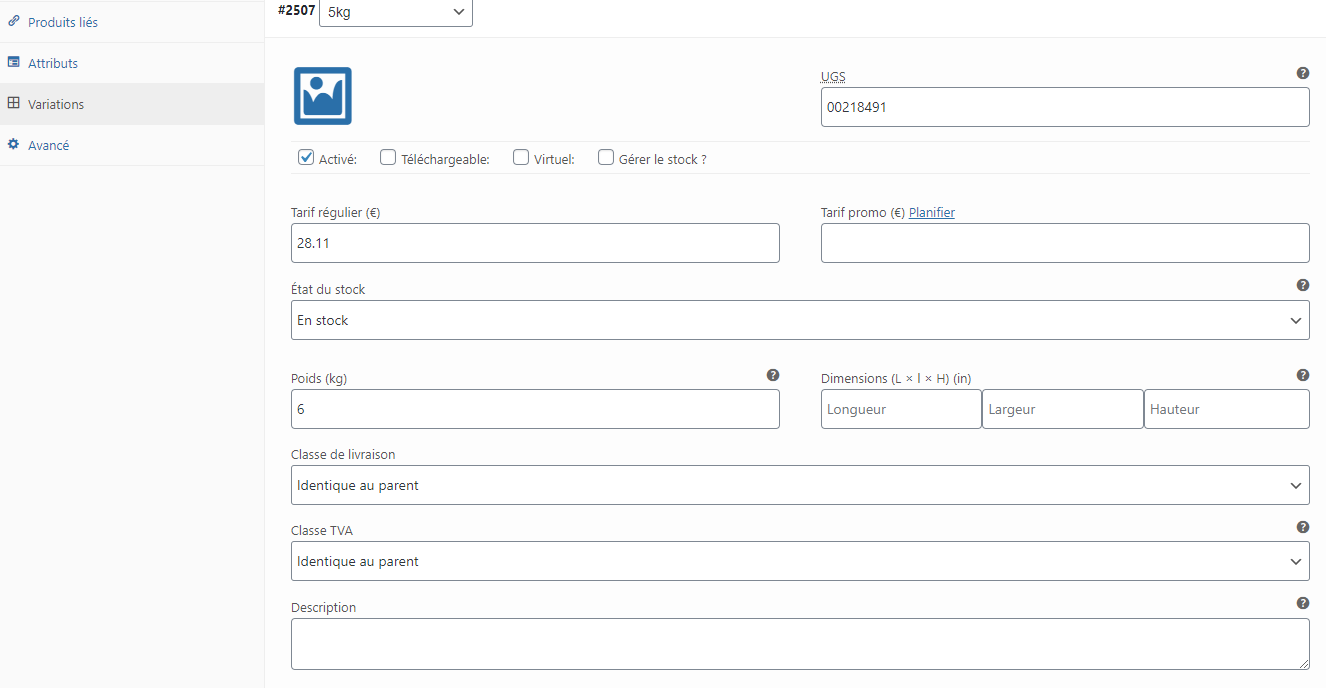


Exemple de différents attributs crées pour ce site.

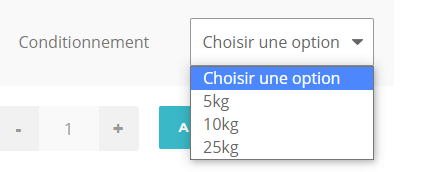
L’utilisation de ces attributs se fait directement dans la page du produit. J’ai dû sélectionner l’attribut correspondant au produit, ici le conditionnement de galets de chlore, pour ensuite utiliser les différents termes correspondants aux types de conditionnement du produit. Ici des sceaux de 5,10 et 25kg.



En spécifiant que ces attributs sont utilisés pour les variations de produit, je peux ensuite configurer les données pour chaque produit. En effet, chaque variation possède différentes données, à commencer par le prix, le poids ou encore la référence du produit, qui seront des données très importantes lors de l’expédition de ces derniers.

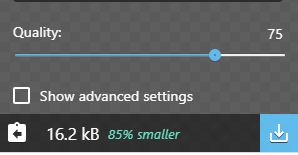


Données d’un produit variable, ici 5kg.



Une fois toutes les données rentrées, l’utilisateur peut choisir le type de conditionnement directement sur la page du produit.

Sur chaque produit, il a évidemment fallu rajouter une image correspondante à ce dernier. Toutes ces images sont le plus souvent disponibles sur le site officiel du produit. Cependant, ces images ne sont, la plupart du temps, pas optimisées et bien trop lourdes. Ceci à une incidence sur la rapidité de chargement des pages du site internet, et donc sur la qualité de l’expérience des utilisateurs. Pour cela, j’ai dû compresser les images des produits avec une application web qui se nomme **Squoosh**. Cet outils proposé par GOOGLE permet d’optimiser les images en gardant une qualité maximale.



Par exemple, pour le produit rentré précédemment, j’ai pu réduire le poids de la photo de 85%, la faisant passer de 108Kb à 16.2Kb. Sur une page contenant beaucoup d’articles de la boutique, ce changement n’est clairement pas négligeable, et a donc été utilisé sur tous les articles créés sur le site. Ainsi, même les utilisateurs ne disposant pas d’une bonne connexion internet pourront charger les pages du site rapidement.

Lors de l’ajout d’une image, la chose importante à faire est de fournir un texte alternatif, permettant d’afficher une description du produit si l’image ne peut pas se charger pour quelconque raison. Ce texte permet aussi d’aider des personnes malvoyantes à connaitre l’intitulé de l’image. Cette petite description correspond à la balise « Alt » dans la balise « img » d’un code HTML.



Rendu d’un produit variable sur la page du produit.

## 2.4 Ajout d’une catégorie de produit

J’ai différencié les produits de deux manières différentes. En les classant par catégorie et par marque.

Pour les catégories, j’ai simplement dû en ajouter dans l’onglet « Catégories » du plugin WooCommerce. Pour rajouter des marques, je me suis servi du plugin « Perfect Brands for WooCommerce » qui permet d’en créer, et de générer des pages pour chaque marque, qui est accessible en cliquant sur le logo de cette dernière présent dans la sidebar\* des différentes pages de produit. Enfin, le choix d’affectation de catégories et de marque se fait directement dans la configuration du produit sur Wordpress.

## 2.5 Ajout de nouvelles caractéristiques sur le produit

Afin d’étoffer la configuration des produits dans WooCommerce, deux options s’offrent à nous, qui sont l’ajout de caractéristiques directement par le fichier ‘Functions.php’, et l’ajout par un plugin nommé « ACF ». J’ai personnellement utilisé les deux méthodes sur deux sites différents, cela m’a permis d’expérimenter les deux façons de faire.

### 2.5.1 Ajout par le code

Un client voulait ajouter des produits sous forme de pack d’heures, afin d’effectuer des interventions. Dans la configuration du produit, nous devions donc pourvoir rajouter une caractéristique, servant à spécifier le nombre d’heures correspondantes au produit.

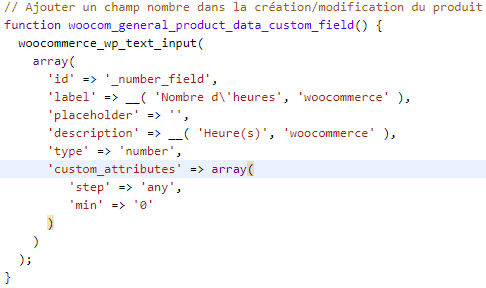
Pour se faire, j’ai dû ajouter un « Hook ». Il nous permet de modifier ou encore d’ajouter du code sans modifier les fichiers principaux. Il en existe deux types :

Les hooks d’action qui permettent d’exécuter une fonction lors de l’utilisation d’une classe spécifique.

Pour se faire, dans mon fichier « Functions.php », je rajoute cette ligne de code :

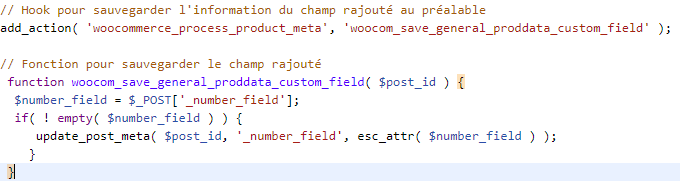


Dans cette dernière, je demande d’ajouter une action lors de l’utilisation de la classe « woocommerce\_product\_options\_general\_product\_data », qui est d’utiliser la fonction « woocom\_general\_product\_data\_field », que je vais déclarer juste après.



La fonction utilisée dans le Hook me permet donc d’ajouter un champ sur la page de configuration de produit. La création de ce champ se fait en utilisant une fonction native de WORDPRESS, contenant un tableau, reprenant toutes les caractéristiques de la balise « input\* » en HTML. Il m’a ensuite fallu ajouter une nouvelle fonction pour sauvegarder les données rentrées dans ce champ lorsqu’on enregistre les informations.

Pour cela j’ai rajouté un nouveau Hook lors de l’utilisation d’une nouvelle classe, demandant l’exécution de la fonction suivante :



Ainsi, la fonction me permet de sauvegarder les données envoyées par l’administrateur par la méthode $\_POST qui, en PHP, permet de transmettre des données sans qu’elles ne s’affichent dans l’URL, augmentant la sécurité lors de la récolte d’information. Ces données sont stockées dans une variable afin de rendre le code plus clair. Enfin, si le champ n’est pas vide, la fonction update\_post\_meta est utilisée.

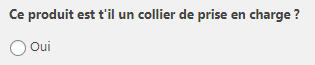
Cette dernière prend en compte l’identifiant du post, ici du produit, le champ à mettre à jour où à enregistrer, ainsi que les informations à envoyer, qui sont ici les données stockées dans notre variable $number\_field. Cette fonction permet donc de faire une requête SQL qui va stocker les informations dans la base de données.

### 2.5.2 Ajout de champs au travers d’un plugin

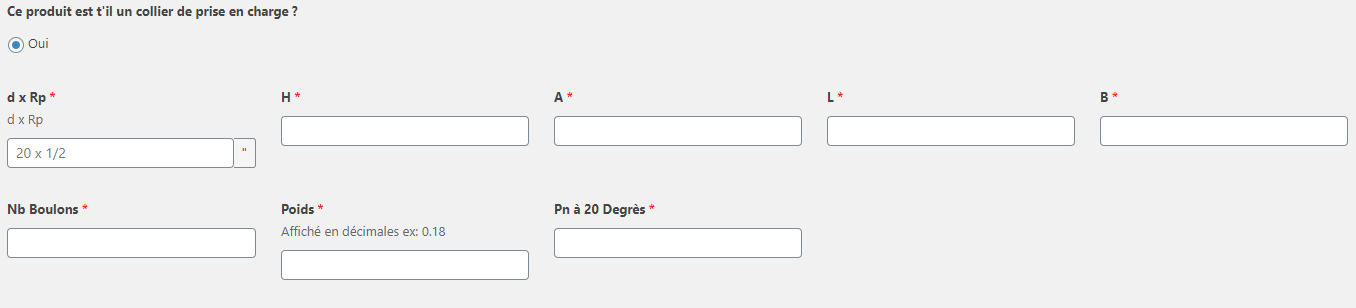
Lors de l’ajout de produits sur le site d’un client, j’avais besoin de renseigner un certain nombre d’informations seulement sur un certain type de produits.

Une autre méthode plus simple m’a permis de rajouter de nouveaux champs. Pour cela, j’ai installé le plugin « ACF ADVANCED CUSTOM FIELD ». Ce dernier, proposant une interface graphique claire, m’a permis d’ajouter un groupe de champ plus facilement et surtout plus rapidement que si j’avais du tout ajouter en ligne de code.

Ainsi, j’ai pu préciser toutes les informations nécessaires, et utiliser une logique conditionnelle pour les afficher où non sur la page de configuration de produit, afin de ne pas polluer cette interface si jamais nous voulons rajouter un autre type de produit. De plus, ces champs seront nécessaires et devront donc être tous renseignés pour effectuer la sauvegarde du produit.



Cette logique conditionnelle est basée sur l’ajout d’un bouton radio\* directement dans la page de configuration de produit qui, s’il est coché, fera apparaitre tous les champs en question. Cette dernière se configure directement dans le plugin.



## 2.6 Affichage des informations du produit dans la description

Les informations listées précédemment sont rentrées dans le but de les afficher sur la page d’un type de produit. Pour cela, j’ai dû trouver le fichier « short\_description.php » disponible dans le les dossiers du plugin WooCommerce.

### 2.6.1 Accéder aux fichiers WooCommerce

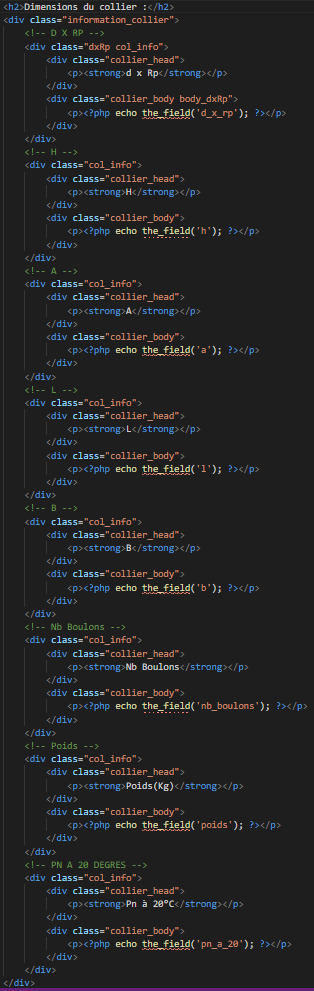
Afin d’accéder aux fichiers je me suis servi du logiciel Filezilla, qui m’a permis de récupérer les fichiers du site qui sont sur le serveur. Pour se faire je me suis servi des informations FTP fournies par l’hébergeur du site. Ces informations contiennent le nom de l’hébergeur, un nom d’identifiant ainsi qu’un mot de passe.

Une fois le bon fichier trouvé, je le télécharge, et recréé la même arborescence de fichier dans mon thème enfant\*. Cela me permettra de garder toutes les modifications faites lors d’une future mise à jour du plugin, car tous les fichiers sont mis à jour à ce moment-là. Ainsi, lors du chargement d’une page, le plugin vérifiera d’abord si le fichier est présent dans le thème enfant, pour l’afficher en priorité, sans quoi il affichera le fichier présent directement dans le plugin.

Arborescence des fichiers dans le plugin. Arborescence des fichiers dans le thème enfant.

Dans le fichier correspondant, j’ai donc rajouté toutes les informations du produit. Pour se faire, j’ai utilisé les langages HTML et CSS.



En premier lieu, une balise H2 décrit les informations qui vont suivre. Je n’ai pas mis de balise H1 car ce n’est pas le titre principal de la page, et qu’il ne peut apparaître qu’une seule fois, afin de respecter les règles de référencement. Cette règle nous permettra d’apparaitre en haut des recherches google et donc de potentiellement faire venir plus de personnes sur le site.

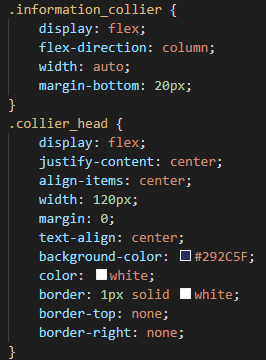
J’ai ensuite créé plusieurs « Div\* » au sein d’une même « Div » afin d’utiliser les « Flexboxs ». C’est une fonctionnalité présente dans le langage CSS et qui permet de positionner plus facilement les éléments sur une page.

Ainsi, les éléments prendront la forme d’un tableau, sans utiliser la balise « table » du langage HTML.

En effet, cette structure est plus flexible si l’on veut modifier la disposition des informations dans le futur, et est beaucoup plus facile à adapter à tous les types d’écrans.

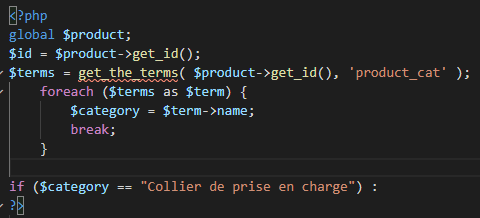
Enfin, pour afficher les informations voulues, j’ai utilisé la fonction « the\_field() » prenant en compte le nom du champ du produit que l’on veut ajouter. Cette fonction est disponible avec le plugin utilisé pour rajouter les champs sur les configurations de produit.

Les classes données à mes balises vont me permettre de cibler mes balises afin d’ajouter des lignes de code CSS pour personnaliser le contenu, comme la couleur de fond, du texte, la taille des divs etc...



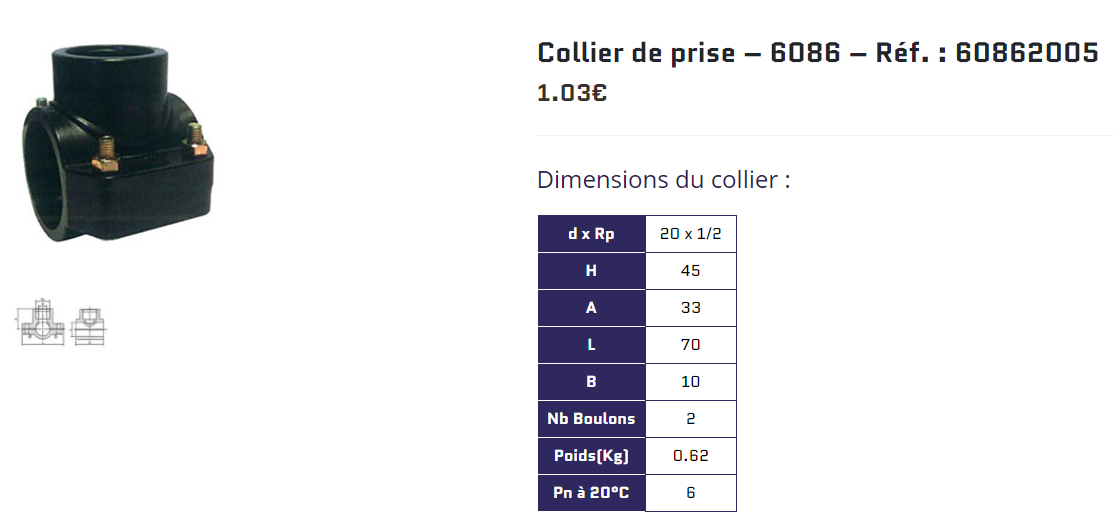
Dans ce code CSS, je sélectionne le nom des classes que j’ai donné précédemment dans mon code HTML, pour ensuite indiquer toutes les modifications que je souhaite y apporter.

Les modifications ayant été faites dans le fichier correspondant aux descriptions des produits, je ne veux cependant qu’elles n’apparaissent que lorsque ce type de produit est affiché. J’ai pour cela créé une nouvelle catégorie de produit, et émis une condition afin que ces données ne s’affichent que lorsque le produit fait partie de cette catégorie. Cette condition, faite en langage PHP englobe dont tout mon code HTML présenté précédemment.



Code permettant de récupérer le nom de la catégorie du produit.

J’ai donc effectué des tests sur deux produits différents. L’un possédant la catégorie « Collier de prise en charge », et l’autre ne la possédant pas. Ainsi, le tableau contenant ces informations devra s’afficher selon les bonnes conditions.

Le tableau s’affiche bien Le tableau ne s’affiche pas

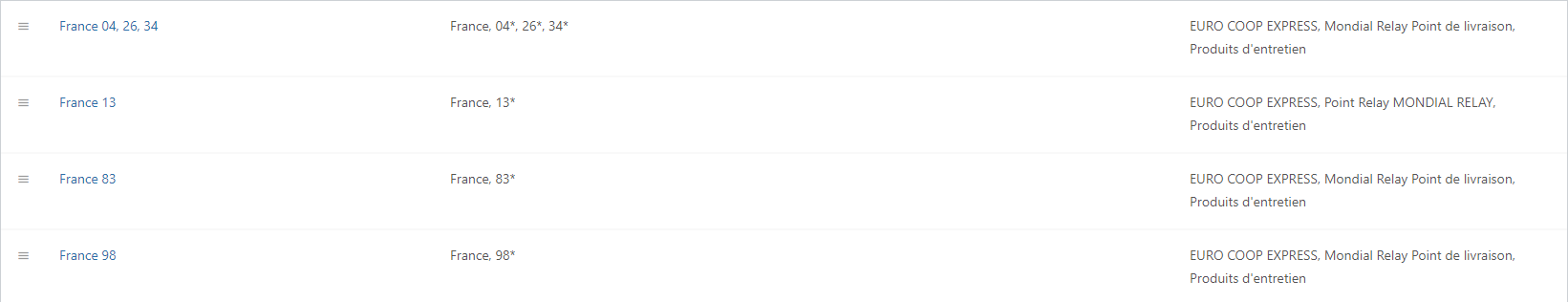
## 2.7 Configuration des méthodes de livraison

Les méthodes de livraison doivent répondre à plusieurs besoins :

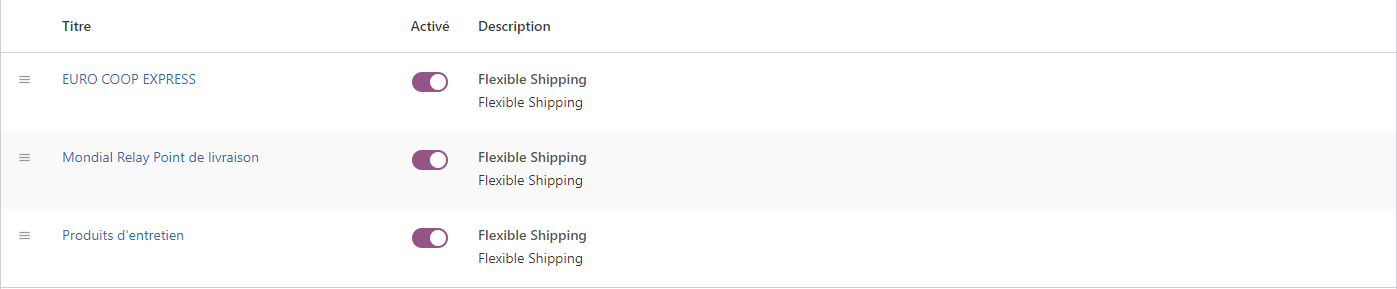
* Dans le cas où le colis ne dépasse pas 30 kg, ne contient pas de matières dangereuses, et est livré en France, le client doit avoir le choix entre deux méthodes de livraison différentes.
* Si les colis excèdent trente kilos ou contiennent des matières chimiques, un seul transporteur agréé doit être proposé comme méthode de livraison.
* Les colis demandant une livraison vers la Belgique ou le Luxembourg doivent proposer la livraison en point Relay et la livraison à domicile, mais ne peuvent pas être livré si le colis contient des produits chimiques.
* Enfin, Les colis de moins de trente kilos et ne transportant pas de matière dangereuse doivent pouvoir être expédiés dans trois pays qui sont la France, la Belgique et le Luxembourg.

Les deux transporteurs ayant des tarifs différents selon les régions dans laquelle les colis sont livrés et selon les tranches de poids de ces derniers, beaucoup de conditions rentrent en compte. J’ai donc téléchargé le plugin « Flexible Shipping », qui permet de configurer plusieurs méthodes de livraisons par régions, tranche de poids et type de produit, tout en intégrant une logique conditionnelle afin de pouvoir proposer le bon type de transporteur.

J’ai donc commencé par ajouter les différentes régions de livraison, dans lesquelles j’ai rajouté trois méthodes de livraison, qui sont Mondial Relay, Euro Coop et Euro Coop produits chimiques, qui contiendra les grilles tarifaires à proposer si des produits dangereux sont présents dans le panier du client.



Différentes zones de livraison.



Les méthodes de livraison au sein des différentes zones.

Enfin, dans chaque méthode de livraison, j’ai rentré les différents tarifs en fonction des tranches de poids. Tout en précisant dans les premières grilles tarifaires que si le produit contient la classe de livraison « Produits chimiques », elle doit se référer à la grille tarifaire correspondante à ces types de produit pour ne proposer que celle-là aux clients.



Exemple d’une grille tarifaire, Ici celle du transporteur EuroCoop.

### 2.7.1 Test de transporteurs au sein de différentes commandes

J’ai réalisé une série de tests afin de voir si les conditions dans mes grilles tarifaires fonctionnent.

Dans un premier temps, j’ai mis dans mon panier un produit ne contenant pas de classes spécifiques, en livraison vers la France.



**Les deux méthodes de livraison sont bien proposées.**

J’ai ensuite mis dans mon panier des articles sans classe de livraison spécifique mais dépassant les trente kilos, toujours en Livraison vers la France.



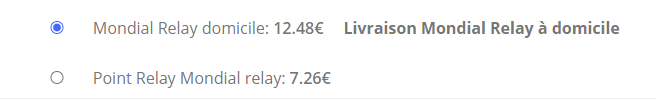
**Un seul transporteur est proposé.**

Le prochain test consiste à mettre deux types de produits, un avec une classe de livraison « Produit chimique », et l’autre sans classe de produit particulière, en livraison vers la France.



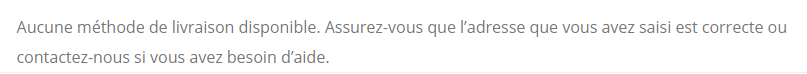
**Le transporteur agréé matières dangereuses est bien le seul proposé.**

Ensuite, des produits sans classe de livraison, demandant une livraison vers la Belgique ou le Luxembourg.



**Les deux méthodes de livraison, à domicile et en point Relay sont bien proposées.**

Le dernier test consiste à demander la livraison de produits chimiques vers la Belgique où le Luxembourg.



**Aucune méthode de livraison n’est proposée, les produits chimiques ne pouvant pas être livrés vers ces pays.**

## 2.8 Installation d’un module de paiement

Le client à voulu passer par le module de paiement MERCANET. Pour cela j’ai simplement dû télécharger le plugin correspondant dans les extensions de WORDPRESS.

Une fois le module installé, j’ai juste eu à rentrer les informations transmises par la banque dans les réglages de ce dernier, et à sélectionner les méthodes de paiement voulues par le client toujours au sein des réglages.

J’ai dans un premier paramétré le module en mode test, afin de procéder à des essais de paiement avec des cartes de test fournies par la banque fournissant le module de paiement. Chaque paiement validé doit ensuite générer une facture. Pour cela, je me suis servi d’un plugin se chargeant de cette tâche. Au sein de ce plugin, J’ai pu configurer le format de la facture, ainsi que toutes les informations nécessaires à la création de cette dernière. Notamment l’affichage de la TVA qui est obligatoire ou encore le nom et l’adresse de l’entreprise.

Pour cela, j’avais au préalable saisis les prix de mes produits TTC en prenant soin de préciser dans WooCommerce qu’ils ont été saisis de cette manière, le plugin déduisant automatiquement la taxe appliquée sur le produit. Enfin, pour calculer la taxe sur les frais de livraison, j’ai rajouté un taux de TVA de 20% applicable sur les expéditions dans les réglages de WooCommerce.

## 3 Modifications au sein des Templates WooCommerce

### 3.1 Modification de la Titlebar

Sur les pages correspondantes aux catégories de produits ou encore aux différentes marques, j’ai voulu personnaliser le titre présent dans la bannière, afin qu’il s’adapte à ces derniers.

Pour se faire, j’ai utilisé le même principe que pour personnaliser la description courte des produits, qui est de récupérer le bon fichier présent dans les templates mais cette fois dans les fichiers de WORDPRESS, et de recréer la même arborescence de fichier dans mon thème enfant, la configuration se faisant normalement directement dans les options de thème. Au sein du fichier « titlebar.php », j’ai donc appliqué les modifications suivantes.

J’ai tout d’abord créé un tableau dans lequel toutes les voyelles sont listées à l’intérieur. Cela va me permettre de proposer deux types de phrase en fonction de la première lettre de la catégorie de produit afin que cela reste dans un français correct.

Je me suis ensuite servi du type de page généré, afin de savoir ce que je veux afficher. Pour se faire je regarde dans l’inspecteur de mon navigateur la classe appliquée à la balise « Body » de ma page. En effet, si la page correspond à une marque de produit, la balise possèdera une classe « tax-wp-brand », et si cette page correspond à une catégorie de produits, la balise possèdera la classe «tax\_product\_cat ».

Pour émettre des conditions au sein de mon code, certaines fonctions natives au sein de WORDPRESS sont disponibles, me permettant de vérifier quel type de classe est appliqué.

$voyelle = array('A', 'E', 'I', 'O', 'U', 'Y', 'a', 'e', 'i', 'o', 'u', 'y');

            // On créé un tableau avec les voyelles

     if (is\_tax('pwb-brand')) {

        $current\_brand = get\_queried\_object();

           echo '<h1>Nos produits de la marque ' . $current\_brand->name . '</h1>';

        } elseif (is\_tax('product\_cat')) {

                        $current\_category = get\_queried\_object();

                         if($current\_category->name == 'Produit d\'entretien'){

                            echo '<h1> Nos produits d\'entretien piscine </h1>';

                        }

      // Si la première lettre du nom de current category ne contient pas de voyelle

                        elseif(!in\_array($current\_category->name{0}, $voyelle)) {

                            echo '<h1> Nos produits de ' . $current\_category->name . '</h1>';

       // Si la première lettre du nom de current category contient une voyelle

         } elseif (in\_array($current\_category->name{0}, $voyelle)) {

                 echo '<h1> Nos produits d\' ' . $current\_category->name . '</h1>';

                        }

                    } elseif (is\_post\_type\_archive('product')) {

                        echo '<h1>Nos produits en vente</h1>';

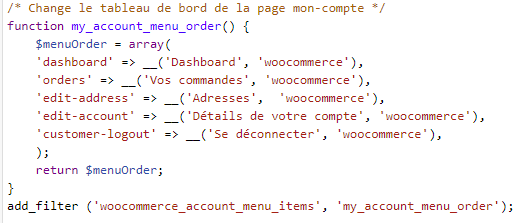
                    } else { ?> <div class="title-h1"><?php echo esc\_html($title); ?></div>

                    <?php } ?>

Ainsi, le titre « H1 » de la bannière s’adapte au type de page.

### 3.2 Modification de la page « Mon compte »

Certaines informations présentes dans la page « Mon compte » n’étaient pas nécessaires, comme l’onglet « Téléchargement », la boutique ne vendant aucun produit téléchargeable. Pour changer cela, j’ai de nouveau utilisé un Hook, mais cette fois ci de type « filter ». Ce dernier me permet de modifier les données d’une fonction déjà présente dans une classe, au lieu de la rajouter comme avec les Hooks de type action.



Cette fonction me permet de redéfinir les données de mon menu , sous forme de tableau dans ma variable « myOrder », afin de retirer l’onglet « téléchargement ».

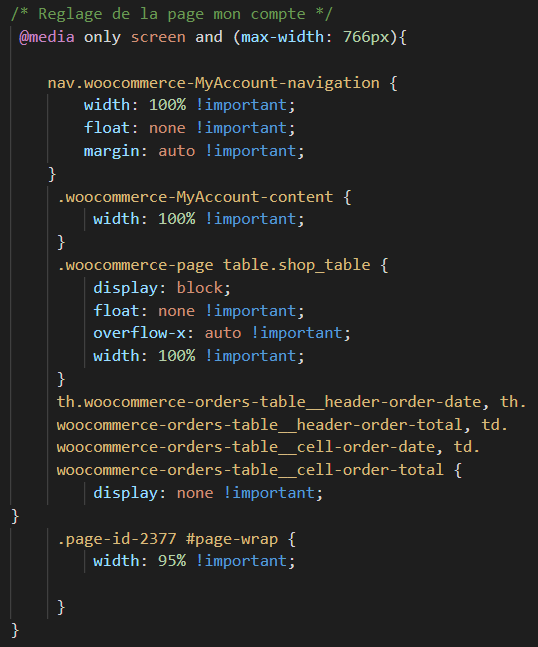
Afin de modifier l’apparence de ce menu, j’ai ajouté des lignes de code CSS, notamment pour rajouter des icones sur la gauche de chaque lien. La propriété « content » m’a permis de réaliser cela, en sélectionnant les liens de navigation grâce à la classe CSS appliquée par le plugin WooCommerce et en précisant que les icônes doivent apparaitre avant le lien, en utilisant le sélecteur « :before ».



Exemple d’une icône rajoutée en CSS.

### 3.3 Utilisation des médias Queries

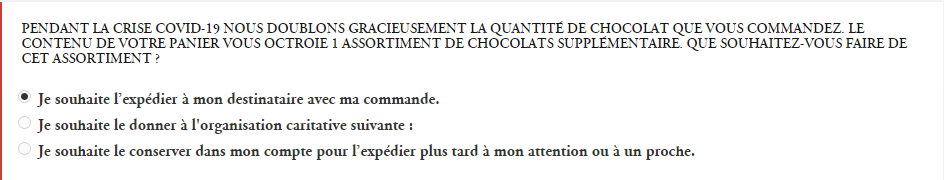
Afin que la page « mon-compte » puisse s’adapter à toutes les tailles d’écran, je me suis servi des « Médias Queries », disponibles dans le langage CSS et permettant de changer des valeurs en fonction de la taille d’écran des utilisateurs.



La valeur « !Important » n’est pas la façon optimale pour modifier les données CSS, en effet, dans la plupart du temps, le fait d’avoir une grande précision dans les sélecteurs CSS permet de modifier les valeurs correspondantes. En revanche, dans ce cas précis, j’ai voulu modifier des données déjà implantées dans un plugin rendant la tache plus ardue. C’est pour cela que cette balise est utilisée dans ce cas.

## 4 Utilisation de Javascript

Dans le cadre d’une modification visuelle sur le site d’un client, j’ai été contraint d’utiliser le langage Javascript, ce langage me permettant de créer des interactions directement entre l’utilisateur et le navigateur, sans rechargement de page. Le client voulait changer l’esthétique de ses boutons radio\* lors d’une option choisie par le client pour valider sa commande.



**Visuel avant la modification**

La spécificité de ce formulaire est que lorsque le client choisis la deuxième option, une liste d’hôpitaux doit apparaître pour que le client puisse en choisir un dans le cadre d’une opération commerciale en rapport avec le Covid19. Le client veut aussi garder la charte graphique du site, notamment le petit bandeau rouge présent.

Pour cela, j’ai d’abord créé le visuel de ce formulaire en HTML et CSS, en prenant le soin d’appliquer la valeur « display : none » me permettant de na pas afficher la liste des hôpitaux qui est cachée lors du chargement de la page, étant donné que la première option doit être cochée au chargement de cette dernière, en utilisant la valeur « checked » sur mon input\* correspondant.

<div class="formType">

        <div class="list-button">

            <div class="button button1 active double-choco col-md-4 col-lg-4"> <input type="radio" checked visibility:

                    hidden name="choix1" id="radioButton1">Please send these complimentary assortments to my gift

                recipients with my order.</div>

            <button class="button button2 hospital col-md-4 col-lg-4"><input type="radio" visibility: hidden

                    name="choix1" id="radioButton2"> Please send them to the following hospital or charity:</button>

            <button class="button button3 save-choco col-md-4 col-lg-4"><input type="radio" visibility: hidden

                    name="choix1" id="radioButton3">Please save them in my zChocolat account for now, I will redeem them

                at a later time (no later than one year from today).</button>

        </div>

   <div class="typehospital">

     <form action="" class="chooseHospital">

        <div class="hospital"> <input type="radio" checked name="hospital" id="newyork"><img src="img/flag233.gif"  style="width:20px;height: 15px;margin-left:10px;" alt="" srcset=""> <label for="newyork">New York Health + Hospitals </label>

                    <p style="font-weight:normal;"> - NYC Health + Hospitals is the largest public health care system in

                        the United States, providing care to more than one million New Yorkers every year.</p>

                </div>

          <div class="hospital"> <input type="radio" name="hospital"> <img src="img/flag233.gif"

                        style="width:20px;height: 15px;margin-left:10px;" alt="" srcset=""><label id="marie"

                        for="">Marie Curie West Midlands</label>

                    <p>- Marie Curie West Midlands provides care and support to people with terminal illnesses and their

                        families.</p>

            </div>

       <div class="hospital"> <input type="radio" name="hospital"> <label for="">Montreal</label>

                </div>

      </form>

        </div>

    </div>

Code HTML du formulaire.

J’ai ensuite stocké mes éléments HTML dans des variables\* Javascript, grâce aux méthodes « getElementById() » et « querySelector() ». La première méthode me permettant de sélectionner mes éléments grâce à leurs identifiants créés au préalables dans la balise HTML correspondante, et le second me permettant de les sélectionner à l’aide de leurs classes.

Par la suite, j’ai du créé des évènements, qui m’ont permis de décrire certaines actions lors d’interactions spécifiques avec mes éléments HTML.

### 4.1 Extrait d’un site Anglophone dans le cadre d’une recherche

Pour la réalisation de mes évènements Javascript, j’ai dû effectuer une recherche sur un site anglophone. J’ai choisi cet extrait sur le site [www.w3schools.com](http://www.w3schools.com), ce dernier étant une référence dans le monde du développement.

### Extrait anglophone

The addEventListener() method attaches an event handler to the specified element.

The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

You can add many event handlers of the same type to one element, i.e two "click" events.

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

The addEventListener() method makes it easier to control how the event reacts to bubbling.

When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

You can easily remove an event listener by using the removeEventListener() method.

Syntax

element.addEventListener(event, function, useCapture);

### Traduction de l’extrait

La méthode addEventListener() relie un évènement sur un élément spécifique.

La méthode addEventListener() relie un évènement sur un élément sans avoir à réécrire du code déjà existant.

Vous pouvez ajouter plusieurs évènements sur un élément.

Vous pouvez ajouter plusieurs évènements du même type sur le même élément, comme deux evènements « click ».

Vous pouvez ajouter des évènements sur tous les objets, pas seulement sur des éléments HTML, comme l’objet window.

La méthode addEventListener() permet de contrôler facilement comment l’évènement réagis au bubbling.

Lorsque l’on utilise la méthode addEventListener, le Javascript est séparé du côde HTML, pour une meilleure lisibilité et vous permet d’ajouter des évènements lorsque vous ne contrôlez pas le code HTML.

Vous pouvez facilement annuler des évènements en utilisant la méthode removeEventListenere().

Syntaxe :

Element.addEventListener(évènement, fonction, capture) ;

J’ai donc ajouté des évènements sur tous les éléments surs lesquels je voulais des interactions.

button1.addEventListener('click', hidden);

        button1.addEventListener('click', addBorder1);

        button1.addEventListener('click', inputCheck1);

        hospital.addEventListener('click', notHidden);

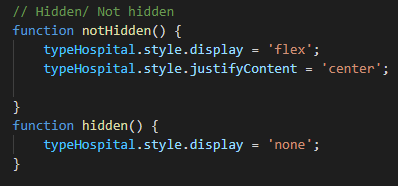
        button2.addEventListener('click', addBorder2);

        button2.addEventListener('click', inputCheck2);

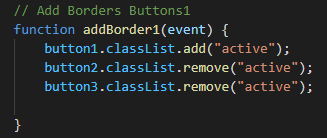
        button3.addEventListener('click', hidden);

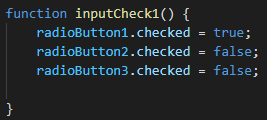
        button3.addEventListener('click', addBorder3);

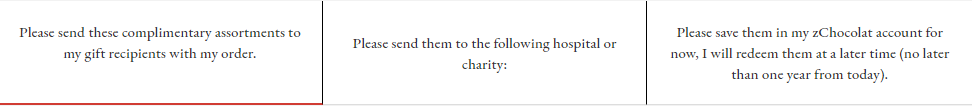
        button3.addEventListener('click', inputCheck3);



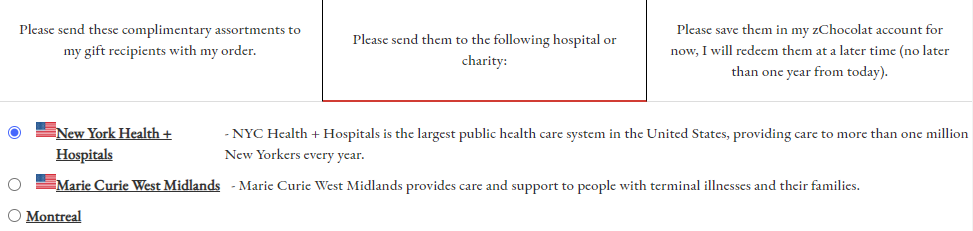
**Les fonctions « hidden » et « notHidden » me permettent de faire apparaitre ou non la liste des hôpitaux.**

Les fonctions « addBorder » me permettent de faire apparaitre la bordure rouge sur l’élément sélectionné. Et cela en ajoutant ou en enlevant la classe « active » de mes éléments, cette dernière ajoutant une bordure sur le bas de mes éléments, générée grâce au langage CSS.

Enfin, les fonctions « inputCheck » me permettent de sélectionner le bouton radio correspondant à mon élément, qui est invisible grâce à la valeur « visibility : hidden ».



Rendu des nouveaux boutons.



Rendu des boutons en sélectionnant la deuxième option.

# 5 Ajout de données utilisateurs lors d’une commande

Certains produis devaient pouvoir ajouter un nombre d’heures d’intervention sur le profil d’un utilisateur. Pour cela, j’ai une nouvelle fois utilisé un Hook, cette fois ci lorsque le paiement d’une commande est complété.

Afin de récupérer les informations de la commande en question, j’ai utilisé une fonction native à WORDPRESS, prenant en compte l’identifiant. Idem pour l’identifiant de l’utilisateur. Enfin, une boucle « for each », est créé, permettant d’effectuer certaines manipulations pour chaque produit acheté, en récupérant le nombre d’heure associé à chaque produit ainsi que le nombre d’heure disponible sur le compte client. Pour enfin additionner ces derniers et d’instancier une requête SQL grâce à la fonction intégrée dans WORDPRESS.

function so\_payment\_complete($order\_id){

  // On récupère les infos de la commande en question

  $order = wc\_get\_order( $order\_id );

  $order->get\_id();

  $order->get\_order\_key();

    $user = $order->get\_user\_id();

  $items = $order->get\_items();

  // Pour chaque item dans la commande

  foreach ( $items as $item ) {

                  // On récupère l'id du produit

                  $product\_id = $item->get\_product\_id();

                  // On récupère le nombre d'heure associé au produit

                  $hours = get\_post\_meta($product\_id, '\_number\_field', true);

   // On récupère les heures enregistrées sur le profil de l'utilisateur

         $old\_hours = (int)get\_user\_meta($user, '\_userHours','meta\_value' );

         $hours += $old\_hours;

   // Requête pour envoyer le nouveau nombre d'heures sur le compte du client

                  update\_user\_meta( $user, '\_userHours', (int)$hours );

              }

}

add\_action( 'woocommerce\_order\_status\_on-hold', 'so\_payment\_complete' );

Fonction permettant d’ajouter des heures au compte client.

Ainsi, si l’utilisateur de possédais pas d’heures au préalable, un nouveau champ est créé dans la table des métas utilisateurs. Au contraire, si le champ existe déjà, une simple update sera exécutée.

# 6 Utilisation d’un Framework PHP

Dans le cadre d’un projet personnel, j’ai utilisé le Framework LARAVEL, qui est open source et utilise, comme le CMS WORDPRESS, l’architecture Modèle-Vue-Controller, et donc la programmation orientée objet.

## 6.1 L’architecture MVC

Cette architecture permet de mieux organiser son code, en le séparant en trois parties principales :

**Les modèles** gèrent les données du site, permet de récupérer des données de bases des classes, pour ensuite les envoyer vers les contrôleurs afin de gérer ces données dans différentes fonctions.

**Les contrôleurs** permettent de prendre des décisions selon ce que l’on veut afficher ou non sur la page. Ce dernier permet par exemple de déterminer si un utilisateur est autorisé à consulter une page ou non, ou encore de transmettre des variables dans les vues afin d’afficher certaines informations récupérées dans une base de données par exemple.

**Les vues** sont composées principalement de code HTML, et contient la partie visuelle de la page. Du code PHP peut aussi être utilisé afin d’afficher les informations contenues dans les variables que le contrôleur à envoyé

Afin d’installer le Framework, j’ai tout d’abord du installer le gestionnaire de dépendance COMPOSER. L’installation s’est faite directement via le terminal, en exécutant les différentes disponibles dans la documentation de ce logiciel.

Ensuite, l’installation de LARAVEL s’est aussi faite grâce au terminal, en exécutant la ligne de commande suivante :



Avant de commencer le codage de la première page du site, j’ai tout d’abord dû réaliser une maquette avec l’outil Figma. Cette dernière me permettra d’avoir un aperçu de ma page, et de faire des modifications visuelles plus rapidement que si je codais ma page au fur et à mesure.



Aperçu de la maquette

## 6.2 Codage de la maquette

Sur LARAVEL, le code des pages visibles par les utilisateurs se fait directement dans le fichier « ressources/views. Au sein de ce dernier, j’ai pu séparer dans différents fichiers les parties de mes pages. Par exemple, j’ai commencé par créer un fichier « app.blade.php » qui m’a servi de structure, dans lequel j’ai pu mettre la base de mon code HTML. J’ai dans ce dernier codé les parties qui seront identiques dans toutes mes pages comme le header. Dans ce fichier j’ai aussi pu préciser à quel endroit se situe la section dans laquelle je vais inclure le contenu de mes autres pages grâce à la fonction « @yield(‘content’) ».

## 6.3 Envoi de données dans ma base de données

Le site aura pour but de proposer des recettes de pâtisserie. Ce dernier devra contenir une page regroupant toutes les dernières recettes postées sur le site. J’ai donc créé un formulaire permettant de créer de nouvelles recettes, me permettant d’alimenter ma base de données. Ainsi, j’ai créé une fonction dans mon fichier « RecipeController » me permettant de renvoyer vers ma page de création qui est la suivante :

@extends('app')

@section('content')

<div class="container">

    <div class="recipe" style="display: flex; flex-direction:column; align-items:flex-start; justify-content:center;">

        <h2>Publier une nouvelle recette</h2>

        {{-- Formulaire permettant d'enregistrer une nouvelle recette --}}

   <form action="{{route('recipe.store')}}" method= "POST" id="recipe-form">

            @csrf

 <div class="form-group">

   <label for="title">Titre de la recette</label>

   <input type="text"name="title" id="title">

   <label for="description">Donnez une petite description de votre recette !</label>

                <input type="text" name="description" id="description">

            </div>

{{--On inclus le formulaire permettant de rajouter de nouveaux champs à notre recette--}}

            @livewire('multiple-fields')

            <hr>

            <button type="submit" class="connexion">Créer la recette</button>

        </form>

    </div>

</div>

@endsection

Pour afficher ma vue, j’ai dû configurer une route pour y accéder, dans le fichier routes/web.php.

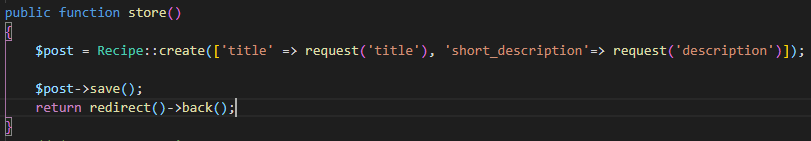


Ici, la route ayant comme URL « /recettes/creation », exécutera la fonction « create » présente dans mon Controller « RecipeController.php ». Enfin, j’ai donné un nom à cette route, qui me servira lors de la configuration de mon lien dans mon code HTML, ici « recettes.creation ».



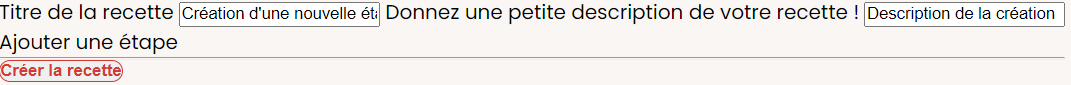
Ainsi, sur ma page de création, je précise l’extension de mon fichier, puis dans quelle section de ce dernier le code doit apparaître. En l’occurrence la section « content » qui correspond au code « @yield(‘content’) » écrit précédemment.

J’ai ensuite créé un formulaire, avec comme action « route(‘recipe.store’) » qui est une route que j’ai créé et qui me permet de renvoyer la fonction « store() » de mon Controller. Cette fonction va me permettre d’enregistrer les données envoyées via le formulaire. Voici le contenu de ma fonction « store() » :



Cette dernière me permet de créer un nouvel objet de ma classe « Recipe », en stockant les informations de mon formulaire dans les bons champs de ma table. Enfin, la fonction « save() » me permet de sauvegarder mes informations dans ma base de données.

La fonction « redirect()->back() » me permet de rediriger l’utilisateur vers une page, en l’occurrence vers la page précédente.



Formulaire de création de recette



Informations stockées dans ma base de données.

# 7 Utilisation des diagrammes UML

Afin de réaliser une nouvelle fonctionnalité, je me suis servi du langage de modélisation UML. Ce dernier comprend une série de diagrammes permettant de modéliser les fonctionnalités d’une application web, et d’ainsi avoir une vision plus ou moins détaillée de ce que l’on doit coder en fonction des différents diagrammes.

J’ai personnellement réalisé trois sortes de diagrammes :

* Le diagramme de cas d’utilisation
* Le diagramme de séquence
* Le diagramme de classes participantes

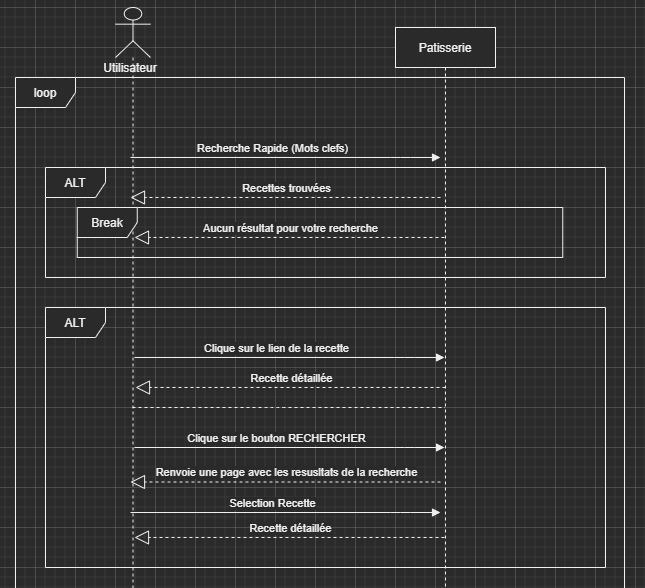
Ces diagrammes ont pour but de modéliser la fonctionnalité de barre de recherche au sein de mon site.

## 7.1 Le diagramme de cas d’utilisation

Ce diagramme a pour acteur l’internaute consultant la page principale du site, voulant effectuer une recherche de recette.

# 

## 7.2 Le diagramme de séquence système



Ce diagramme sert à détailler les requêtes entre l’utilisateur et le système lors d’une fonctionnalité précise. Cette dernière étant toujours la barre de recherche, plusieurs réactions sont attendues en fonction des résultats de la recherche et des manipulations de l’utilisateur.

Ainsi, lors de l’utilisation de la recherche rapide, le système devra soit afficher les résultats de la recherche en dessous de la barre, soit préciser qu’aucun résultat n’a été trouvé. L’utilisateur devra avoir deux possibilités, qui sont de cliquer sur le lien qui est proposé dans les résultats en dessous de la barre de recherche, ou bien d’appuyer sur le bouton « ENTER », qui lancera le chargement d’une page avec les résultats de la recherche. Si aucun résultat n’est trouvé, ceci sera précisé dans la page de résultats.

## 7.3 Le diagramme de classe participantes

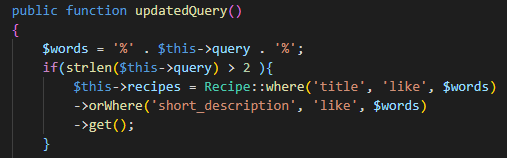
Ce diagramme a pour but de montrer les classes qui devront être réalisées lors du codage de la fonctionnalité de barre de recherche. Chaque classe contient dont des attributs et des méthodes, sous forme de fonctions, et toutes les classes sont reliées avec celles dont elles ont besoin pour fonctionner, pour par exemple récolter des informations dans une base de données afin de les afficher.

# 8 Livewire

Pour la réalisation de la barre de recherche, je me suis servi du framework de Laravel Livewire. Ce dernier permet d’implémenter des fonctionnalités JAVASCRIPT très rapidement et ce en continuant de coder avec le langage PHP. Après l’installation via php artisan, j’ai donc créé une classe via le terminal, avec la commande « php artisan make :livewire Search ». Cette commande m’a permis de créer automatiquement une vue ‘Search.blade.php ‘ me permettant d’afficher les informations voulues ainsi qu’un Controller ‘Search.php ‘ me permettant d’envoyer toutes les informations nécessaires à ma vue.

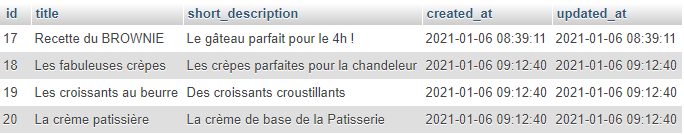
J’ai aussi créé une autre vue se nommant ‘searchResult.blade.php’ qui me servira de page d’affichage des résultats des recherches.

Ensuite, dans mon Controller, j’ai tout d’abord déclaré deux variables. L’une étant une chaine de caractère vide qui me servira à stocker la requête de l’utilisateur, et l’autre étant un tableau vide qui se remplira des informations récoltées dans la base de données. J’ai aussi créé trois fonctions. La première se nomme « render() » et me permettra de renvoyer la vue contenant ma barre de recherche, la deuxième, « updatedQuery() », me permet l’affichage des résultats de la recherche en temps réel sous cette dernière.

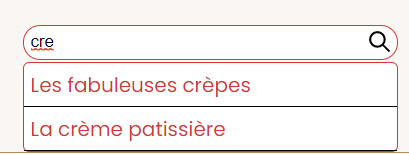


Ainsi, la variable « $words » me permet de stocker la requête de l’utilisateur, entourée de pourcentages. Ces derniers vont me servir par la suite dans ma requête SQL. En effet, si la requête de l’utilisateur est supérieure à deux caractères, je demande de rechercher dans ma base de données et plus précisément dans ma table ou mes objets « Recipe » sont stockés, les données ressemblant à la requête de l’utilisateur, dans le titre des recettes ou bien dans les descriptions courtes de ces dernières.

Afin de réaliser des tests sur cette fonctionnalité, j’ai rajouté des informations dans ma base de données via un formulaire que j’ai réalisé au préalable. Ainsi, dans ma table « recipes », j’ai les informations suivantes :



Donc en tapant les lettres « cre » dans ma barre de recherche, les résultats devraient être « Les fabuleuses crêpes » et « La crème patissière ».

 Cette fonction me permet donc de renvoyer les résultats en temps réel.

Si aucun résultat n’est trouvé, ceci est bien spécifié comme précisé dans le diagramme de séquence système.

La page de résultats lorsque l’utilisateur appuie sur le bouton « ENTER ». Les deux résultats sont bien présents.

# 9 Ajout d’un système d’authentification

Afin d’ajouter cette fonctionnalité, je me suis servi d’un Framework de Laravel, qui se nomme Fortify. Ce dernier permet d’ajouter facilement un système d’authentification tout en permettant de personnaliser facilement l’aspect graphique des pages d’inscription ou de connexion. L’installation se fait en ligne de commande grâce à « Composer ». L’installation génère de nouveaux fichiers, d’une part dans mon dossier « App/Actions/Fortify », possédant toutes les classes permettant la création de nouvel utilisateur, la validation de mot de passe, la réinitialisation de mot de passe, le changement de mot de passe ou encore le changement d’information de l’utilisateur. D’autre part, dans le fichier « FortifyServiceProvider.php », plus précisément dans la fonction « Boot », un certain nombre de fonctions sont présentes, me permettant de savoir quel fichier « vues » utilise les différentes fonctionnalités. C’est à partir de cela que je sais comment nommer mes fichiers « vues » qui vont me permettre d’intégrer mes formulaires d’inscription ou de connexion.

# 7 Utilisation de l’outil GIT

Lors de cette formation j’ai été amené à utiliser l’outil de versionning GIT. Celui-ci est un indispensable, surtout lors de la création de projet collaboratif dans lequel plusieurs personnes sont amenées à travailler dessus. Ce dernier permet aussi de créer des versions du projet afin de revenir en arrière si des erreurs ont été faites, ou encore de créer des « branches », qui nous permettent de créer de nouvelles fonctionnalités dans toucher directement au code initial, afin de ne pas « casser » le code déjà présent, pour ensuite fusionner la branche créée avec la branche principale une fois que le travail effectué est fonctionnel.

En complément de cet outil, j’ai utilisé « GIT HUB », qui m’a permis d’envoyer mon travail dans un dépôt en ligne. Ainsi, une sauvegarde de mes fichiers était toujours présente en ligne, me permettant de la récupérer de n’importe où, et permettant aussi de travailler à plusieurs sur un même projet, chacun pouvant envoyer ses modifications et récupérer des fichiers toujours à jour.

Pour se faire, après avoir initialiser mon projet git grâce à la commande « git init », j’ai pu directement sur le site « GIT HIB » créer un dépôt en ligne. Après cela, j’ai pu rentrer les commandes fournies par l’outil GIT HUB afin de lier mon projet au dépôt en ligne. Enfin, la commande « git push origin master » m’a permis d’envoyer mes fichiers dans mon dépôt.

# Conclusion

Durant les dix mois de cette formation, j’ai appris à rechercher les informations dont j’ai besoin dans cette ressource presque infinie qu’est internet. J’ai cependant rencontré beaucoup de difficultés notamment dans la première partie de cette formation. En effet, apprendre en recherchant soit même les informations n’est pas toujours facile, bien que j’aie pu être aiguillé pour cela, créant en moi une certaine frustration, qui m’a accompagné jusqu’au démarrage de mon stage. C’est surtout ce dernier qui m’a convaincu que cette méthode allait m’être utile tout au long de ma future carrière de développeur Web.

Au fur et à mesure de l’avancée de ce stage en entreprise, des fonctionnalités de plus en plus complexes m’ont été demandées, allant de paires avec l’avancée de mes cours sur l’ensemble des langages présents dans cette formation de Développeur Web et Web Mobile.

Je retiendrai personnellement deux mots de cette expérience de six semaines. L’entraide et le partage.

Au-delà de l’aspect apprentissage des différents langages, ce projet m’a demandé de rechercher de nombreuses informations sur internet afin de trouver des solutions aux différents problèmes rencontrés. Apprendre à chercher est selon moi un indispensable dans ce métier. Cela m’a donc permis de découvrir l’entraide qu’il peut y avoir dans ce domaine. En effet il jouit d’une communauté soudée, prêtre à répondre aux questions des différents utilisateurs. Cet esprit d’entraide s’est aussi fortement ressenti au sein de ma promotion. Certaines personnes ayant un niveau plus avancé n’ont jamais hésité à venir en aide aux personnes étant plus en difficulté.

C’est ainsi que j’espère vous avoir exposé mes compétences acquises visant à devenir un développeur Web.

# Remerciements

Je souhaite remercier tout particulièrement mon tuteur de stage, Simon METIVIER, pour son implication au sein de mon stage. Pour sa confiance qui m’a permis de travailler sur des cas concrets dès le début de cette expérience professionnelle, me permettant de découvrir les différents aspects de ce métier. Pour sa pédagogie et sa transparence qui n’ont fait que confirmer que c’est bien le métier que je souhaite faire.

Je remercie également mon formateur Serge BALDACCI qui a su m’accompagner tout au long de ma formation en me guidant vers les bonnes sources nécessaires à mon apprentissage, et qui m’a donc préparé au métier de développeur en m’apprenant à chercher les bonnes ressources aux bons endroits.

Un merci tout particulier à « GRAFIKART » qui à été une de mes ressources principales lors de mon apprentissage, et a toutes les personnes qui mettent à disposition des ressources sur internet afin d’apprendre et de progresser, et cela gratuitement.

Je tiens aussi à remercier La région Provence Alpes Côte d’Azur ainsi que la structure de Pôle Emploi qui m’ont d’une part conseillé cette formation, et d’autre part qui l’ont rendu possible en la finançant.

Je remercie également mes camarades de formation, qui ont su utiliser la bonne mentalité d’entraide tout au long de cette dernière.