Développeur Web et Web Mobile

Dossier de projet

Florian Casterot

Titre Professionnel de niveau 5

DWWM

Septembre 2020 / Juin 2021

Table des matières

[Présentation de l’entreprise 3](#_Toc72832617)

[Présentation de la société : 3](#_Toc72832618)

[Présentation de l’équipe : 3](#_Toc72832619)

[Liste des compétences du référentiel : 4](#_Toc72832620)

[Les projets : 5](#_Toc72832621)

[Présentation du projet front : 5](#_Toc72832622)

[Les outils et langages utilisés : 5](#_Toc72832623)

[Maquettage : 6](#_Toc72832624)

[HTML : 8](#_Toc72832625)

[CSS : 10](#_Toc72832626)

[Responsive : 13](#_Toc72832627)

[Présentation du projet back : 14](#_Toc72832628)

[Les outils et langages utilisés : 14](#_Toc72832629)

[Présentation du chifoumi en Javascript (jQuery) : 15](#_Toc72832630)

[15](#_Toc72832631)

[Création de la base de données : 17](#_Toc72832632)

[Démarrage du projet Symfony : 17](#_Toc72832633)

[Configuration de l’affichage : 19](#_Toc72832634)

[Espace Utilisateur : 19](#_Toc72832635)

[Espace Administrateur : 22](#_Toc72832636)

[Espace Jeux : 24](#_Toc72832637)

[Jeu d’essai : 27](#_Toc72832638)

[Amélioration à apporter au projet : 29](#_Toc72832639)

[Projet Back : 29](#_Toc72832640)

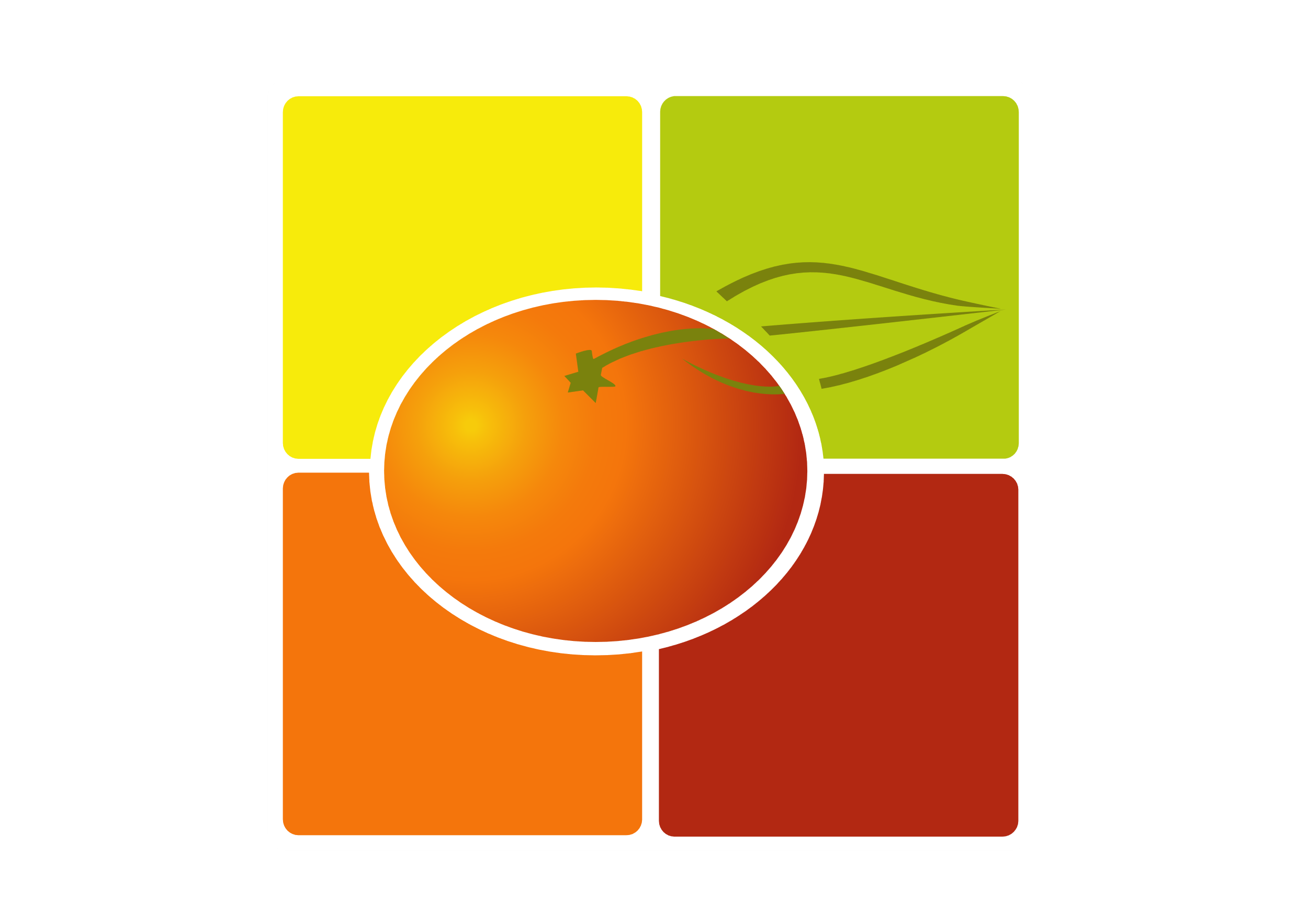
[Extrait du site anglophone : 30](#_Toc72832641)

[Veille technologique sur les vulnérabilités de sécurité : 32](#_Toc72832642)

[Sécurité de base sous WordPress : 32](#_Toc72832643)

[Conclusion : 33](#_Toc72832644)

# Présentation de l’entreprise



## Présentation de la société :

eMandarine :

eMandarine a été créée en 2010 par Alexandre Mauric à Poitiers (86).

eMandarine est une agence web à très forte culture marketing. Créative, réactive dans les domaines de la stratégie de communication et du Marketing digital.

### Présentation de l’équipe :

Les personnels de l’équipe avec qui j’ai travaillé sont de deux développeur et un commercial.

* Alexandre Mauric, gérant et développeur ;
* Delphine Meunier, web designer et intégratrice web ;
* Afis Mounirou, cadre dirigeant, directeur commercial.

## Liste des compétences du référentiel :

Chacune des compétences citées ci-dessous, sont tirée du « Référentiel Emploi Activités Compétences du titre professionnel DWWM ».

1. Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité

* Maquetter une application
* Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

1. Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité

* Développer les composants d’accès aux données
* Développer la partie back-end d’une application web ou web mobile

# Les projets :

## Présentation du projet front :

Le projet demandé par le client été une conception d’un site web avec une partie e-commerce (click & collect).

Nous devons retrouver une page principale synthétise l’ensemble des autres pages et qui fera office de page contact.

Nous avons deux pages pour les actualités (recettes et actualités) ainsi que des pages présentant l’entreprise (GAEC de la Bocagère) et ou les trouver (ferme et foires)

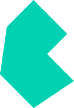
Enfin nous avons une page boutique accessible par des clients particuliers et professionnels (affichage de produit différent).

### Les outils et langages utilisés :

Nous utilisons Wordpress pour avoir un back-office déjà créé pour le client. Nous utilisons le thème de base de eMandarine que j’ai retravaillé pour coller à l’image du client.



J’ai utilisé le plug-in WooCommerce pour la partie e-commerce, que j’ai paramétré pour faire du click & collect.



J’ai utilisé Bulma pour le responsive et la mise en forme ainsi que PHP pour faire quelques petites fonctions.



Pour la création de la base de données j’ai utilisé XAMPP pour MySQL et Visual Studio Code pour coder.

Pour la partie graphique et maquette j’ai utilisé Adobe XD, Adobe Photoshop et Adobe Illustrator.

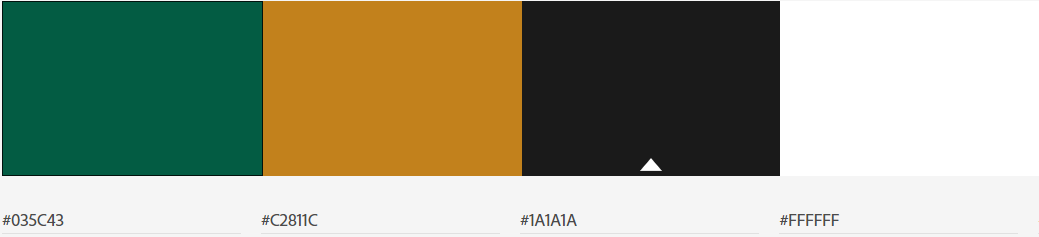


### Maquettage :

Pour la partie du maquettage, les seules consignes données par le client été de garder les couleurs présentes ainsi que l’esprit général des flyers fournis.

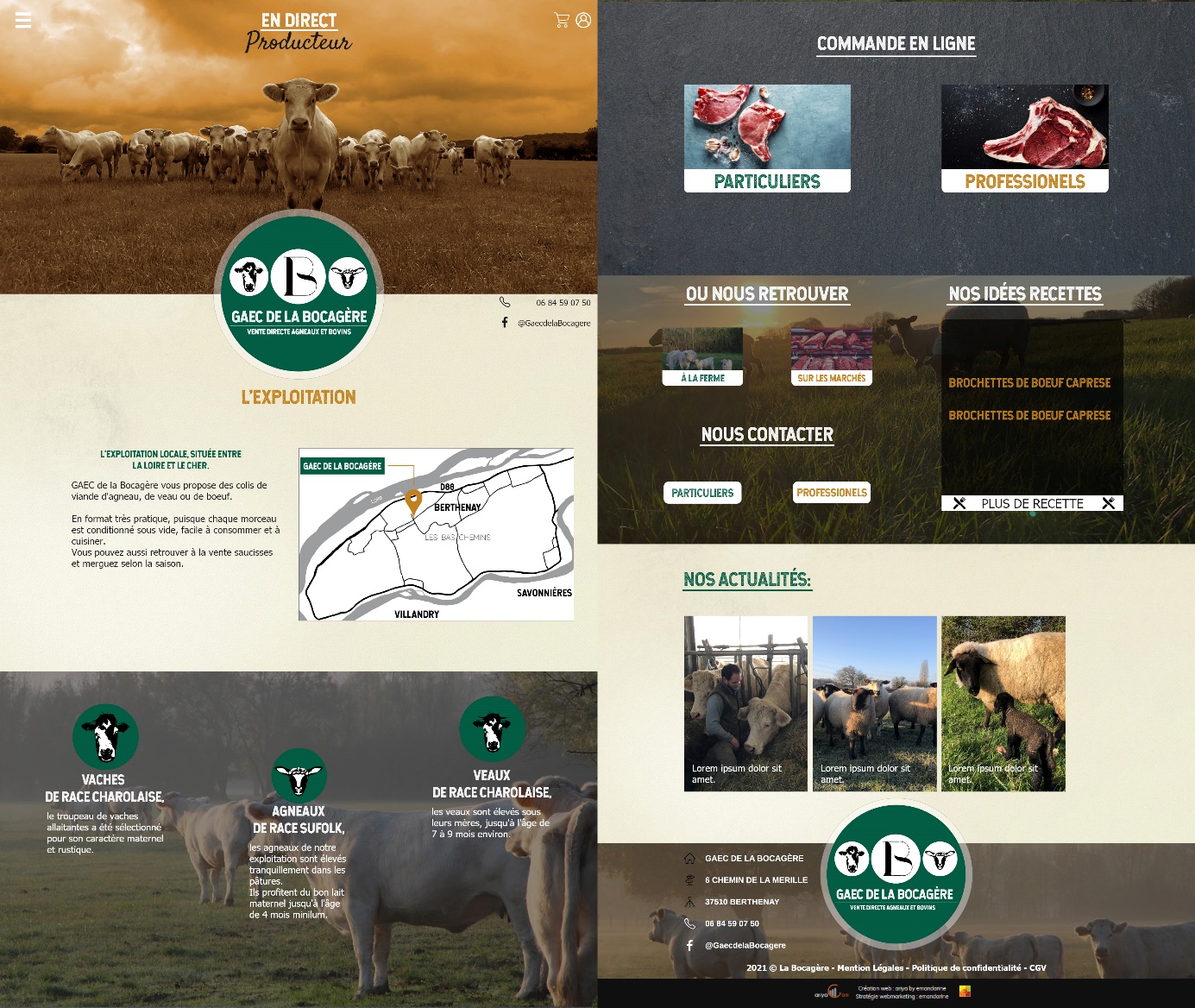
J’ai récupéré les couleurs ainsi que les polices d’écriture avec Adobe Illustrator, puis ajuster le logo.

Les polices utilisés sont « Tahoma » pour le texte, « Hansief Rough » pour les titres et « Satisfy » pour le texte en début de site (header de l’accueil).



Avec Adobe Photoshop j’ai retouché une des photos fournis pour créer l’image qui deviendra le Header.

Enfin j’ai débuté mon maquettage avec Adobe Xd. On ma demandé uniquement un format Desktop pour présenter l’idée au client, mais j’avais déjà en tête a quoi le site ressemblera en format Mobile.



### HTML :

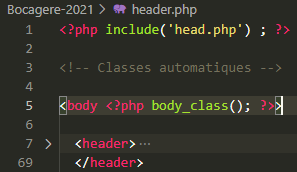
Le HTML (HyperText Markup Language) est utilisé pour créer et représenter le contenu d’une page web et sa structure.

Partant du thème de base de eMandarine, les structures HTML sont déjà en place. Travaillant avec Wordpress, chaque partie est séparé en différents fichiers.

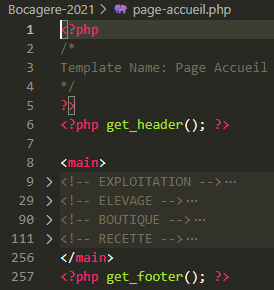
Dans le fichier head.php, nous retrouvons le début des balises structurantes.

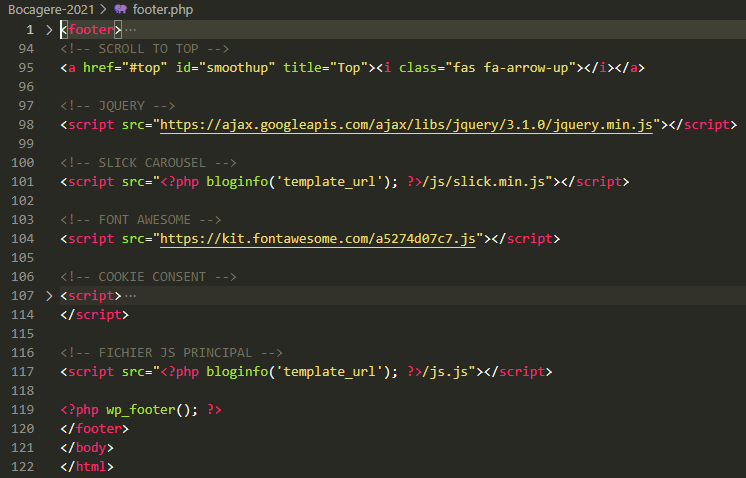


Dans le fichier header.php, nous retrouvons le début du body, ici tout le header est séparé du fichier principal.



Dans le fichier page-accueil.php, nous retrouvons l’intégralité de la page principale.



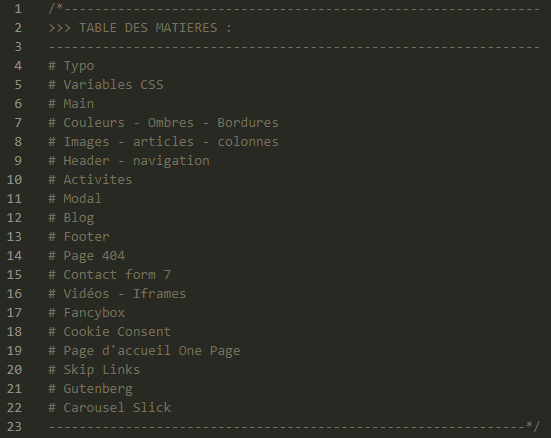
Enfin dans le footer.php, nous avons les fermetures des balises structurantes.

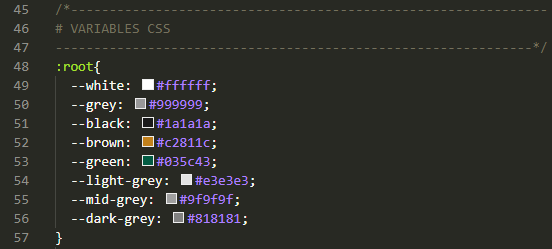
### CSS :

Le CSS (Cascading Style Sheets) ou feuille de style en cascade en français permet de mettre en forme la structuration de la page web faite en HTML.

Pour la mise en forme, eMandarine utilise la bibliothèque CSS Bulma.

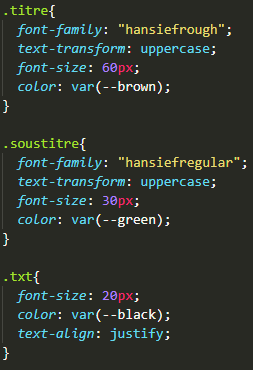
Nous avons donc deux fichiers CSS présent, le fichier minifier de la bibliothèque Bulma ainsi que le fichier template sur lequel j’ai apporté toutes les modifications.

Dans notre fichier template.css, tout est organisé pour une facilité la recherche.

Aussi pour simplifier l’utilisation des couleurs, nous utilisons des variables.

A la manière d’une bibliothèque CSS, je me suis fais des classes réutilisables pour les titres, sous-titres et le texte.

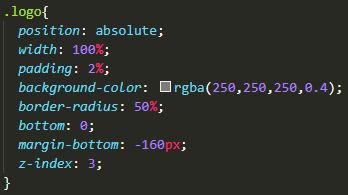
Cela m’évite de surcharger le CSS avec les « fonts-family », « font-size » et « color », je réutilise juste mes classes dans mes balises HTML.



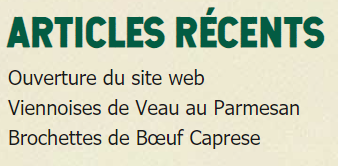
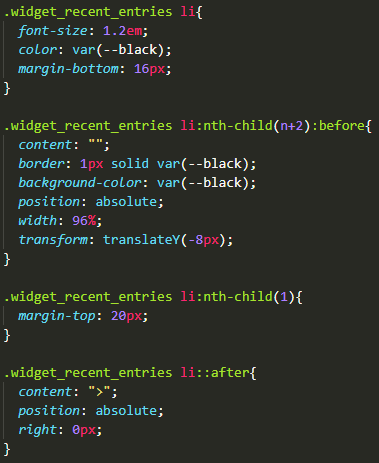
Le positionnement du logo d’après la maquette ma fais utiliser une « position : absolute ;».

Le logo devait être positionner entre le header et ma première section.

J’ai donc donné une « height » de 85vh à mon header pour laisser apparaitre le début de la section suivante.

Ensuite j’ai positionné mon logo en bas de la section avec « bottom : 0 ; » puis je l’ai remonté à l’endroit voulu avec un « margin-bottom » négatif.

Pour refaire l’affichage du widget qui affiche les articles récents, j’ai utilisé les pseudos classes « nth-child », « after » et « before ».



Widget après modification

Widget avant modification

### Responsive :

Pour avoir un site responsive il faut dans un premier temps ne pas oublier d’insérer la balise viewport meta dans le head de notre HTML.

Ensuite pour le colonage j’ai utilisé les balises de Bulma.

La classe « columns » sers à délimiter la ligne.

La classe « column » délimite une colonne.

Le « is-3 » défini la largeur de la colonne.

La classe « is-4-desktop » défini la largeur de la colonne au format Desktop (1024px).

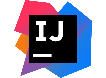
## Présentation du projet back :

Suite à la création d’un chifoumi en JavaScript, j’ai décidé de faire une plateforme ou l’on peut créer un profil et sauvegarder ces scores en base de données.

Un espace utilisateur est accessible, pour voir son profil et ces scores.

Un admin peut gérer la totalité des membres.

### Les outils et langages utilisés :



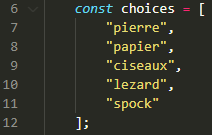
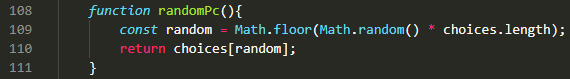
Pour la partie back, j’ai utilisé Symfony. Pour la création de la base de données j’ai utilisé XAMPP pour MySQL et IntelliJ pour coder la partie back.

J’ai utilisé la bibliothéque BootStrap pour la mise en forme du front. Le chifoumi a été créé en Javascript.

### Présentation du chifoumi en Javascript (jQuery) :

Dans un premier temps, j’ai créé une fonction « randomPc » qui fera des choix aléatoires pour l’ordinateur.

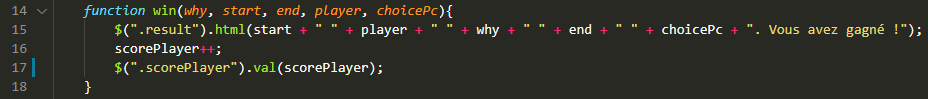
Cette fonction retourne aléatoirement un des cinq choix possible contenu dans mon tableau « choices ».



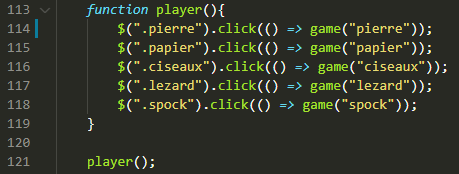
### 

Une fois cette fonction créée, j’ai créé la fonction « game » qui prend comme paramètres « player » qui sera le choix effectué par le joueur.

Grace à un « switch » j’ai listé tous les cas possibles dans le jeu. Chaque cas retourne une fonction qui incrémentera sois le score du joueur, sois celui de l’ordinateur, ou ne fera rien en cas d’égalité.

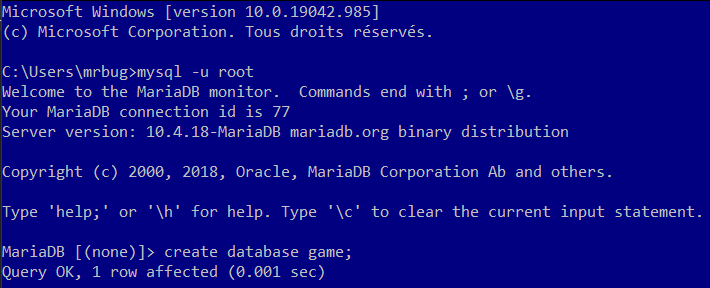
J’ai ensuite créé mes trois fonctions « win », « lose » et « nul » qui me permettrons d’incrémenter les scores ainsi que d’afficher les résultats de la manche (ce qui a été joué et si l’on a gagné ou perdu).

Enfin j’ai créé la fonction « player » qui enregistre le choix du joueur et l’insère dans le paramètre de la fonction « game ». Pour alléger un peu le code j’ai choisi d’utiliser des fonctions fléchées.



### Création de la base de données :

Après avoir démarrer XAMPP, pour créer ma base de données j’ai utilisé l’invite de commandes Windows.



La commande « mysql -u root » me sers à me connecter à la base de données.

Ensuite pour créer ma base de données j’utilise la commande « create database game ; ».

### Démarrage du projet Symfony :

Pour créer mon projet Symfony j’ai utilisé la commande « symfony new game –full ».

Une fois le projet créé, je l’ai ouvert dans IntelliJ, pour me servir du terminal intégré.

Dans un premier temps j’ai paramétré l’accès à ma base de données et le mailer dans le fichier .env.

J’ai ensuite démarré le serveur avec la commande « symfony server:start -d ».

Pour commencer, j’ai créé l’entité User (« symfony console make:user), le formulaire d’enregistrement (« symfony console make:registration-form »), le formulaire de connexion (« symfony console make:auth ») et le formulaire de demande de mot de passe (« symfony console make:reset-password ») avec les commandes suivantes.

J’ai ensuite créé mon entité Game qui enregistrera les scores et nom du jeu. Il y aura une relation avec l’entité User. Ce sera une relation ManyToOne car l’utilisateur peu avoir plusieurs score et jeu mais le score n’appartient qu’à un seul joueur (User).

### Configuration de l’affichage :

#### Espace Utilisateur :

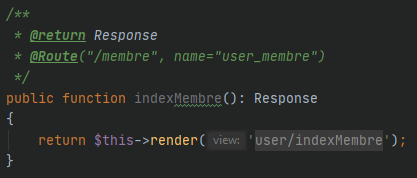
Dans un premier temps j’ai commencé par gérer la partie « Espace membre ».

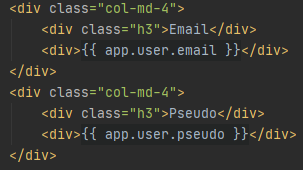
J’ai donc fais un « symfony console make:crud » sur l’entité User.

Le CRUD (Create, Read, Update, Delete), crée automatiquement le UserController, le UserType (formulaire), et la totalité des templates (index, new, show, edit, form, delete).

Je n’ai eu besoin que de trois templates (indexMembre, editProfil et editPass).

Dans le UserController j’ai supprimé les fonctions non utilisables pas un utilisateur lambda (new, show, delete) pour ne garder que la fonction « index » pour afficher le profil et « edit » pour pouvoir le modifier.

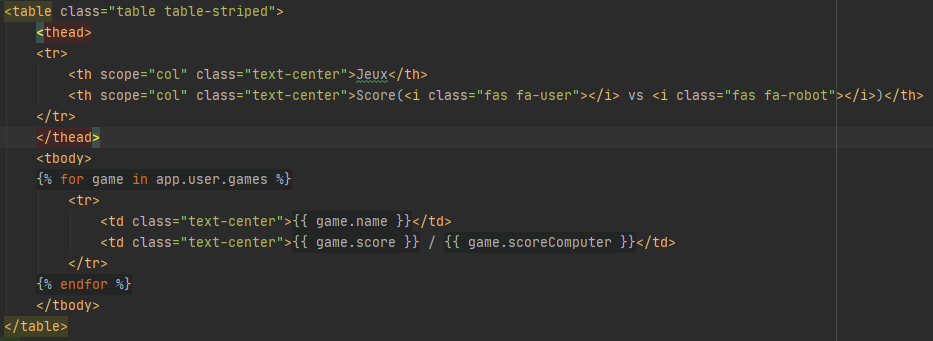
Pour la fonction « index », j’ai seulement gardé le « render » en suppriment le « findAll() » pour n’afficher que les informations de l’utilisateur connecté dans le template « indexMembre ».

Pour afficher les informations correspondantes à l’utilisateur dans le template j’utilise la variable {{ app.user }} qui me donne accès à toute les données inscrites dans l’entité User de l’utilisateur connecté.

J’ai voulu aussi pouvoir afficher les scores des jeux dans l’espace membre.

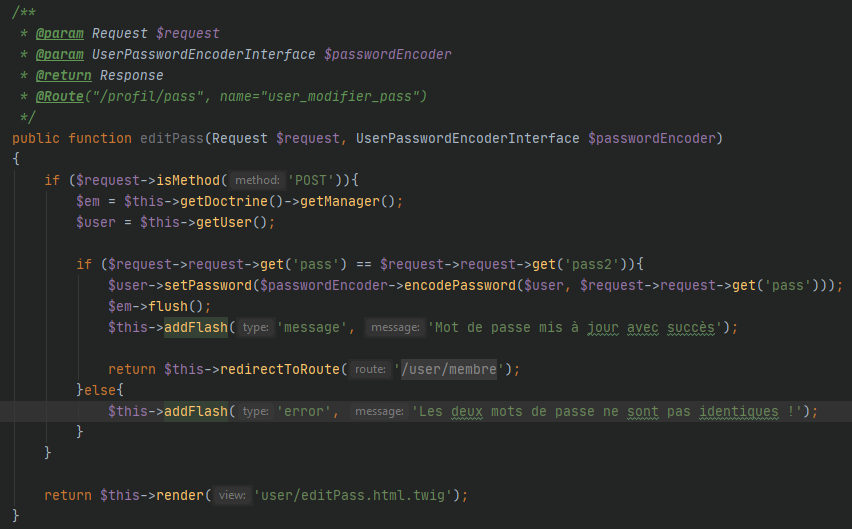
Grace à la relation entre l’entité User et l’entité Game, nous pouvons récupérer les informations stockées.

Pour les afficher dans mon template, je fais une boucle « for » pour récupérer la totalité des scores.



Pour la fonction « editPass » qui me servira a modifier le mot de passe de l’utilisateur, j’ai fait un formulaire directement dans le fichier « editPass.html.twig ».

La fonction vérifie que les deux mots de passes renseignés sont identiques et s’ils le sont, le mot de passe est encodé avant d’être sauvegardé en base de données.

J’ai utilisé la fonction « addFlash() » pour afficher les messages d’erreur ou de succès après la redirection.

#### Espace Administrateur :

Pour mon espace administrateur j’ai créé un « AdminController » avec trois templates (adminIndex, adminShow et adminEdit).

Dans chacune des fonctions de mon « AdminController », je fais des injections de dépendances pour avoir accès à mon entité User.

Ma fonction index à accès au « UserRepository » grâce à une injection de dépendance.

Grace à ça, je peux utiliser la fonction « findAll() » pour récupérer tout les utilisateurs dans mon template.



Les fonctions « show », « edit » et « delete » ont étaient récupérer sur le CRUD du l’entité User.

Pour les utiliser dans « AdminController », j’ai seulement modifié les routes et les « render » de chacune des fonctions.

Pour l’affichage dans le template, j’utilise une boucle « for » ce qui me permet d’afficher la totalité des utilisateurs dans mon tableau. J’utilise la variable {{ user }} suivi de l’intitulé du champ souhaité pour afficher les informations souhaitées.

Pour l’affichage des rôles dans le tableau, l’utilisation d’une boucle « for » est nécessaire car les différents rôles sont enregistrés dans un tableau.

On retrouve dans la dernière case du tableau les trois boutons d’actions. Le bouton « voir » et « modifier » nous amène vers les pages souhaités, tandis que le bouton « supprimer » nous affiche uniquement une fenêtre d’alerte pour confirmer la suppression de l’utilisateur.

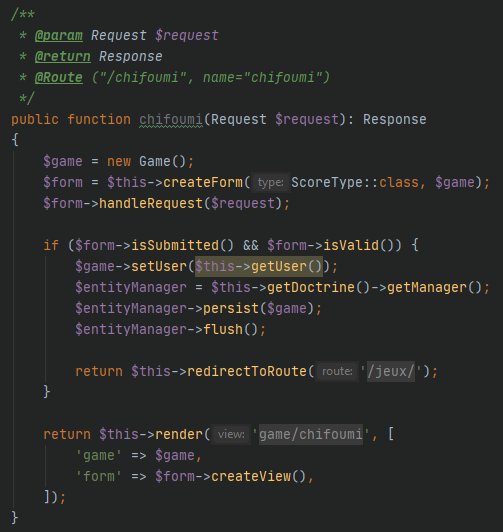


#### Espace Jeux :

Pour la partie jeu, j’ai créé un « GameController » avec une fonction « chifoumi ».

Cette fonction récupère mon formulaire « ScoreType ».

Au moment de la vérification du formulaire, je récupère l’ID de l’utilisateur avant de sauvegarder pour l’insérer dans la table Game pour que pouvoir récupérer les scores du joueur pour les afficher dans l’espace membre.



Mon formulaire « ScoreType » récupère trois « input » et un bouton de validation.

L’input « name » récupère le nom du jeu auquel on joue, l’input « score » récupère le score du joueur et l’input « scoreComputer » récupère le score de l’ordinateur.

Dans le template j’affiche aux endroits voulu mes inputs du formulaire grâce aux fonctions {{form\_start(form)}} et {{ form\_end(form) }}.

Entre ces deux fonctions je peux afficher les champs voulus grâce à la fonction {{form\_row(form.xxx)}}, « xxx » étant le nom du champ correspondant donner dans le formulaire « ScoreType ».

Ici l’input « name » ne sera pas visible car il à la classe « d-none » de BootStrap (display : none ;).

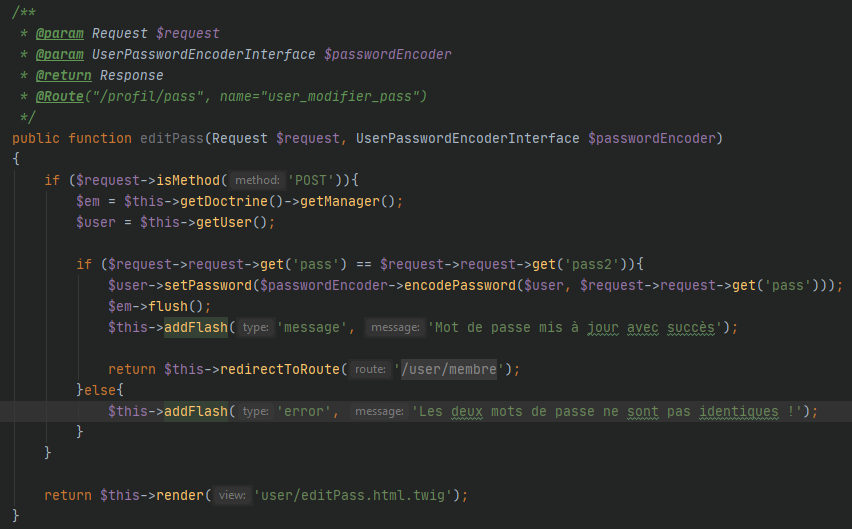
Les deux inputs « score » et « scoreComputer » on une valeur de départ à 0.

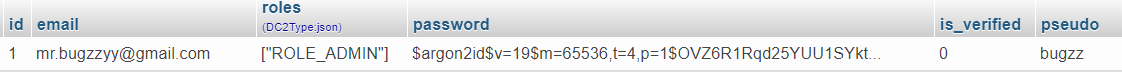
Les scores s’incrémentes grâce à deux fonctions Javascript qui gère le jeu.

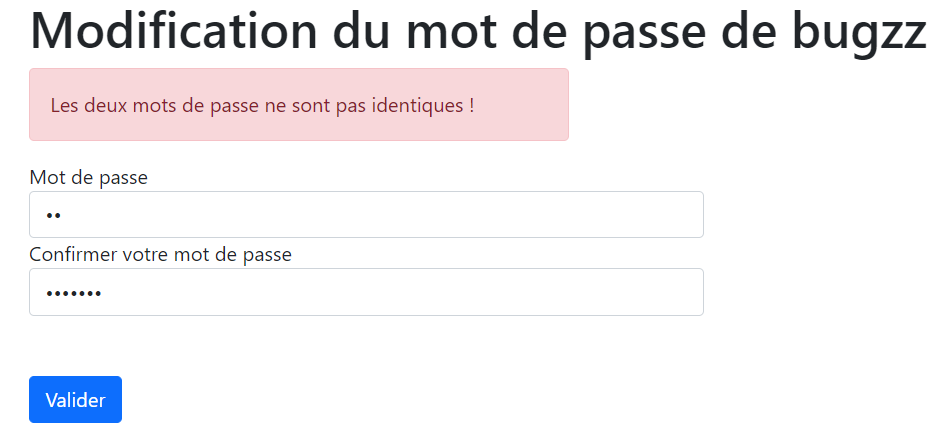


### Jeu d’essai :

Pour le jeu d’essai je présente la fonction permettant à l’utilisateur de modifier son mot de passe.

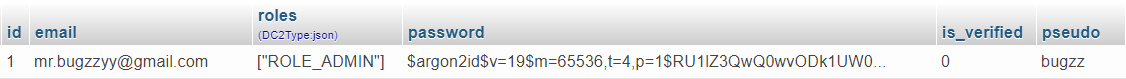
Voici la fonction permettant la vérification des deux mots de passes renseignés.

Ici nous voyons le mot de passe « hasher » dans la base de données.

Si les deux mots de passes renseigné dans les inputs sont différents la page est rechargée, les champs sont vidés et un message d’erreur s’affiche. (Ici les champs ne sont pas vides pour représenter que les deux mots de passes ne sont pas identiques).

Si les deux mots de passes sont identiques, l’utilisateur est redirigé vers son espace personnel et un message de succès est afficher.

Ici nous voyons de nouveau le mot de passe « hasher » en base de données.

On peut constater en comparant avec le mot de passe de base que le « hashage » à bien changé, donc l’opération demandé est réussie.

## Amélioration à apporter au projet :

### Projet Back :

Des améliorations futures sont à venir sur le projet.

Dans un premier temps, j’ajouterai la possibilité à l’utilisateur d’uploader un avatar sur son profil.

Pour ce faire, je compte utiliser le bundle VichUploader.

Pour l’installation j’utiliserai la ligne de commande «  composer require vich/uploader-bundle ». Pour la configuration du bundle j’utiliserai la documentation disponible sur le dépôt GitHub officiel.

J’ai déjà vue qu’une partie « known issues » est disponible pour aider en cas de problème déjà résolu.

Dans un second temps j’ajouterai la possibilité de filtrer les scores en fonction du jeu jouer et de trier les scores des meilleurs au moins bons.

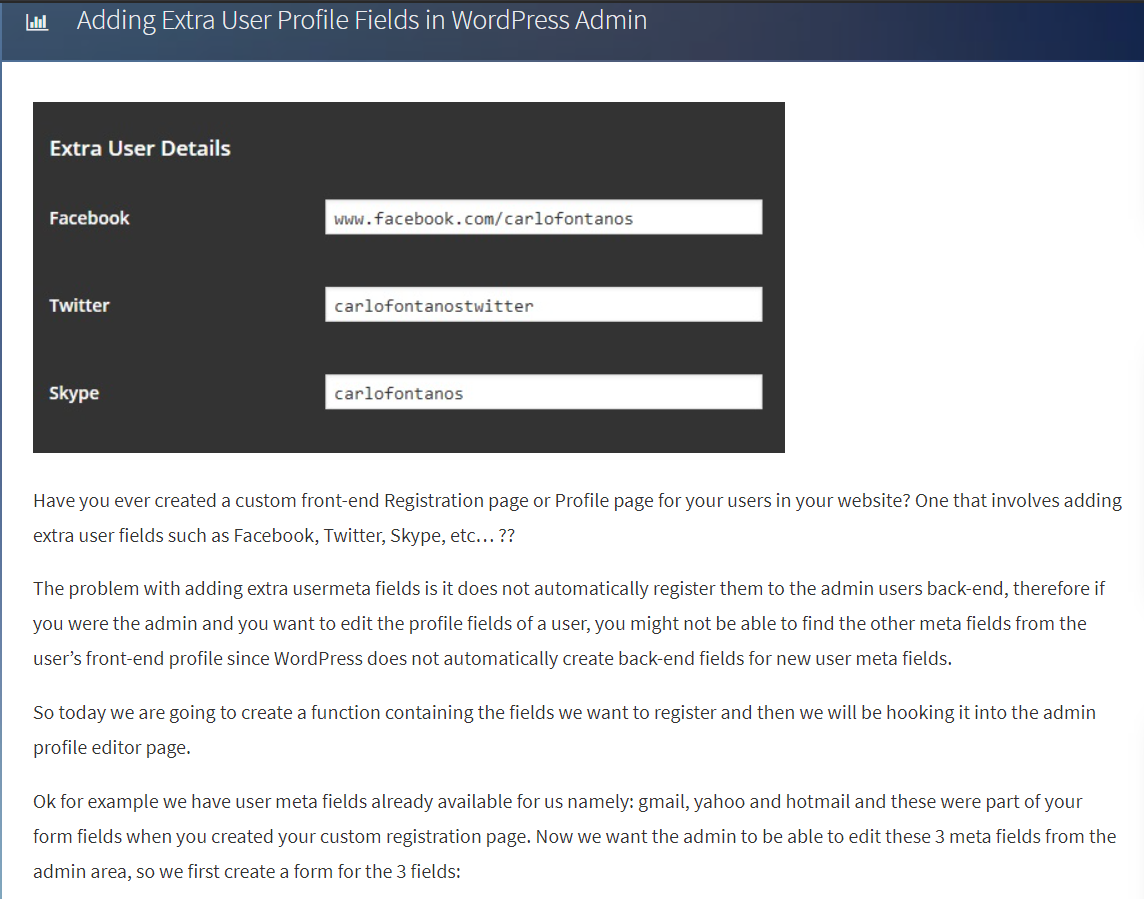
Pour filtrer en fonction du jeu jouer, je ferai un menu déroulant listant les jeux disponibles grâce à un formulaire.

Une fonction dans mon « GameController » me servira pour la recherche.

Je ferai une injection de dépendance dans cette fonction pour accéder au « GameRepository » ce qui me permettra d’effectuer une recherche en fonction des jeux existant.

Bien sûr, la partie front sera améliorer avec des animations des boutons, une meilleure mise en forme et de nouveaux jeux en Javascript.

# Extrait du site anglophone :



Avez-vous déjà créé une page d’enregistrement personnalisé en front-end ou une page de profil pour vos utilisateurs sur votre site web ? Ce qui implique l’ajout de champs supplémentaire tels que Facebook, Twitter, Skype, etc… ??

Le problème avec l’ajout de champs supplémentaire de métadonnées est qu’il n’est pas automatiquement enregistré dans le back-office de l’administrateur, donc si vous êtes administrateur et vous voulais modifier les champs du profil d’un utilisateur, vous ne pourrez peut-être pas trouver les autres champs venant du profil utilisateur front-end car WordPress ne crée pas automatiquement des champs dans le back-office pour les nouveaux champs métadonnées.

Donc aujourd’hui nous allons créer une fonction contenant les champs que nous voulons enregistrer et nous allons les accrocher dans la page profil éditable de l’administrateur.

Ok, pour l’exemple nous avons des champs de métadonnées déjà disponible pour nous, à savoir : Gmail, Yahoo et Hotmail et ceux-là faisait partie du formulaire quand vous avez créé votre page d’enregistrement personnalisé. Maintenant nous voulons que l’administrateur soit capable de modifier les 3 champs de métadonnées depuis la zone administrateur, donc nous créons en premier un formulaire pour ces 3 champs.

D’ailleurs, c’est dans la structure du formulaire donc nous avons le contrôle total des champs personnalisé (comme si vous voulez ajouter d’autre « labels » ou changer le design).

Maintenant pour la partie sauvegarde, nous devons créer une fonction qui s’accrochera au bouton de sauvegarde de la page profil de l’utilisateur, nous utiliserons : « personal\_options\_update » et « edit\_user\_profile\_update ».

Maintenant le problème avec le code au-dessus c’est que tous les utilisateurs auront ces champs dans leur profil back-office. Donc si vous traitez avec des utilisateurs qui ont des rôles et capacités différentes, vous ne voudrez peut-être pas qu’ils aient ces champs. Une solution rapide pour corriger ça est d’utiliser la fonction « current\_user\_can() » de WordPress pour empêcher d’autres rôles de voir ces champs.

Il faut juste entourer votre formulaire avec :

# Veille technologique sur les vulnérabilités de sécurité :

## Sécurité de base sous WordPress :

Dans un premier temps, au moment de la création du site WordPress, pour améliorer la sécurité de la base de données, la première chose à faire est de modifier les préfixes de table.

Ensuite pour la partie formulaire j’ai utilisé les plug-ins Contact Form 7 et Honeypot for Contact Form 7.

Ce dernier permet d’ajouter une fonctionnalité anti-spam. Deux paramètres rentrent en compte pour la vérification de spam.

Honeypot rajoute un champ dans le formulaire (invisible pour l’utilisateur) et s’il est rempli cela signifie que c’est un spam.

Deuxième fonctionnalité, il vérifie en combien de temps le formulaire est rempli, un bot le remplira beaucoup trop vite ce qui permet de le bloquer.

# Conclusion :

Après 9 mois de formation