

***TITRE PROFESSIONNEL DWWM***

DEVELOPPEUR WEB ET WEB MOBILE



***VETOTOIL***

regnier sylvain

***Sommaire***

1. Compétence du référentiel couverte

1.1-Développer la partie Front-End d’une application

1.2-Développer la partie Back-End d’une application

1. Présentation du projet

2.0.1-Méthode Agile

2.1-Spécification Technique

2.2-Description des différents langages utilisés

2.3-Résumé du cahier des charges

2.3.1-Description fonctionnelle des besoins

2.3.2-UML

2.4-Création des Wireframes

2.5-Réalisation des maquettes

1. Introduction et structure
   1. Structure Front-End
   2. Responsive
   3. Validation W3C Validator
   4. Accessibilité
   5. Partie dynamique avec JavaScript

4.Back-End

4.0.1MCD

4.0.2MLD

4.1Squelette Symfony

4.2Base de Donnée

4.2.1Sous Symfony

4.2.2Sous MVC

4.3Création des Entity et controller

4.3.1Sous Symfony

4.3.1.1Modèle physique de donnée sous symfony

4.3.2Sous mvc

4.3.2.1Requete SQL

4.3.3Requete DQL

4.4Exemple d’ajout de bibliothèque(PHPMailer)

4.5Sécurité

4.5.1Les mots de passe utilisateur

4.5.2La sécurisation des données inséré par l’utilisateur

4.5.3Sécurité des routes

5.Conclusion

6.Lexique

**01| Compétences du référentiel couvertes**

*1.1 Développer la partie Front-End d’une application web ou web mobile en intégrant les recommandations de sécurité.*

• LES MAQUETTES

• REALISER UNE INTERFACE UTILISATEUR WEB OU MOBILE STATIQUE ET ADAPTABLE

• DÉVELOPPER UNE INTERFACE UTILISATEUR WEB OU MOBILE DYNAMIQUE

Pour la réalisation du site web « VETOTOIL », j’ai créé des maquettes afin de développer le site. Ensuite, elles ont été rendues dynamiques et fonctionnelles en suivant les bonnes pratiques de développement.

*1.2 Développer la partie Back-End d'une application Web ou Web-mobile en intégrant les recommandations de sécurité.*

En développant la partie back-end du site, j’ai intégré les mesures nécessaires de sécurité pour la protection des données ainsi que contre les attaques malveillantes, telles que la vérification de chaque donnée insérée par l’utilisateur (injections SQL, injections de script, etc.), mais aussi en utilisant des UUIDv7 uniques pour rendre chaque ID aléatoire au lieu d'avoir des séquences prévisibles de chiffres, et le Captcha v3 de Google pour prévenir les inscriptions automatisées par des robots. J'ai également contrôlé l'accès aux données et aux fonctionnalités du site via un système de rôles. La partie back-end présentée est développée avecle framework Symfony, mais une version en Php sans framework mais en utilisant un paradigme d’organisation de code de type MVC. En outre, une fonctionnalité d’envoi de mails à l’administrateur a été mise en place, réalisée avec PHPMailer.

**02 Présentation du projet**

VETOTOIL est une application conçue pour faciliter la prise de rendez-vous chez le professionnel de votre choix, tel qu'un vétérinaire ou un toiletteur (la liste n'étant pas exhaustive), sans nécessiter de recherche en ligne préalable ni de multiples appels téléphoniques souvent sans réponse. L'objectif est de simplifier ce processus en permettant aux utilisateurs de fixer directement une date et une heure de rendez-vous convenant à toutes les parties, directement depuis leur téléphone ou tablette ou ordinateur via le site internet.

VETOTOIL vise à faciliter la prise de rendez-vous en permettant, après inscription sur le site, à l'utilisateur de rechercher les professionnels inscrits dans son secteur. En sélectionnant son domaine d’activité, il peut ainsi consulter les premières disponibilités pour répondre à une urgence ou planifier des rendez-vous à des dates souhaitées. Cela permet de consulter directement depuis un téléphone ou un ordinateur les rendez-vous disponibles. Le patient a également la possibilité de visualiser ses rendez-vous en cours et passés, ainsi que le suivi de traitement de son animal, le cas échéant. De plus, il peut ajouter des animaux à sa fiche, actuellement limités aux chiens ou chats, avec une liste prédéfinie de races pour chacun.

Pour des raisons de sécurité, la société quand a elle pourra s’inscrire, mais il devra forcément avoir un Siret valide et sera soumis à validation par l’administrateur du site. Ensuite il pourra rattacher des employés déjà crée ou en crée des nouveaux, tous cela sera de la responsabilité de la société.

*Le but principal de VETOTOIL est de faire un Doctolib mais pour nos animaux*

*2.0.1 Méthode Agile*

• MÉTHODE AGILE (AGILE SOLO) : Dans le cadre du développement de VETOTOIL, j'ai adopté une approche personnelle de la méthode Agile, en m'assignant des tâches quotidiennes, en planifiant les étapes et en estimant la durée nécessaire pour chaque phase de conception et de réalisation. Chaque samedi, je prenais le temps de faire le point sur l'avancement du projet et de définir les tâches à accomplir pour la semaine suivante.

Voici un exemple :

Date :09 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Faire le header | Faire le header principal  Les liens ne sont pas encore fonctionnels car je n’ai pas les routes pour le moment  Ok pour mobile tablette et pc | 90% |

Date :09 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Faire le footer | Le footer est ok mais comme pour le header les liens ne sont pas mis ok pour mobile tablette et pc | 90% |

Date :09 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Page d’accueil | Faire l’interface de la page d’accueil suivant la chartre graphique les liens ne sont pas inséré et j’ai besoin d’une partie JavaScript ok pour mobile tablette et pc | 80% |

Date :10 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Page d’accueil | Fait le js pour changer le mot vétérinaire en toiletteur avec animation | 100% |

Date :10 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Conception de la page condition générale d’utilisation | Rechercher la loi et faire la page en fonction de la chartre graphique (j’ai utilisé un générateur de condition générale) | 100% |

Date :10 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Faire le RGPD | Idem que la page précédente | 100% |

Date :10 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Faire les condition pour les images | Idem que pour le rgpd | 100% |

Date :10 février 2024

|  |  |  |
| --- | --- | --- |
| Tâche | Etape | %Réalisation |
| Crée la partie user | Création de l’entité registration du controller du formulaire et du twig et faire le fonctionnement | 100% |

**2.1Spécifications Technique**

*Général :*

- Editeur de code : Visual Studio Code

-extension utilisées :

\*Composer for Visual Studio Code : permet de de faire des commandes rapides pour composer

\* IntelliPHP for Visual Studio Code : permet de travailler plus rapidement car l’IA essaie de prédire la suite du code

\*Live Server for Visual Studio Code : permet d’ouvrir une page HTML ou JavaScript en local

\*Prettier Formatter for Visual Studio Code : formatage du code

\*Twig Formatter for Visual Studio Code : formatage du code pour twig

- Outil de versionning : GitHub

- Maquettages : Figma

- Modèle conceptuel et logique de donnée : Looping

- Serveur php : Xampp

- Modele physique de données : Interface visuelle de PhpMyAdmin

**2.2 Projet VETOTOIL**

-Langage et balisage : HTML,Bootstrap et CSS

-Langage de programmation : PHP, Javascript et twig

-Fait sous Symfony

Librairie :

-PHPMailer pour l’envoie de mail (‘https://github.com/PHPMailer/PHPMailer’)

-VichUploaderBundle pour insérer des images(ou fichier) (‘https://symfony.com/doc/current/controller/upload\_file.html’)

-Karser pour faire le RecaptchaV3 de Google pour sécuriser les envoie de mail et d’inscription (‘https://github.com/karser/KarserRecaptcha3Bundle’)

-UuidV7 (fonction unique) pour générer des id aléatoire et « unique » pour certaine base de donnée sensible. (‘ https://symfony.com/doc/current/components/uid.html#working-with-uuids’)

- password-hasher pour crypter un mot de passe pour qu’il ne soit pas récupéré(‘https://symfony.com/doc/current/security/passwords.html’)

**2.3 Cahier des charges**

*Objectif :*

-Etablir un programme utile pour simplifier la prise de rdv chez les professionnels pour les animaux.

Pour les Professionnels :

-Gagner du temps, car il y aura moins d'appels téléphoniques pour la prise/modification/annulation de rendez-vous.

- Possibilité de ne plus avoir de créneaux libres dans leurs plannings.

-Pour les Particuliers :

-Ne plus attendre d'avoir des dates fournies par le professionnel et de vérifier ses propres disponibilités, car les disponibilités seront disponibles immédiatement sur le site internet.

-Avoir un suivi des rendez-vous et chez quel professionnel, ainsi que les traitements dans l'historique.

-Il doit être responsive et s’adapter aux différente taille d’écran (smartphone, tablette, pc)

-Le site aura aussi bien du Back-End et du Front-end

*Fonctionnalités principales :*

*Pour tous :*

Partie inscription user ou tout le monde s’inscrira avec les mêmes information demandé (nom, prénom, email(ne peux avoir deux emails identique), password) ensuite à la connexion ils seront redirigés vers une autre page des complément d’information.

*Pour les Sociétés :*

-création d’une fiche société (information demandé Siret, nom de la société, profession de la société, adresse de la société, complément adresse de la société, code postal de la société, ville de la société, téléphone de la société, téléphone du dirigeant, image société, date de création, date de résiliation, date de validation)

-Doit pouvoir crée un employé (fiche user normale)

-Doit pouvoir rechercher un employé par son email s’il est déjà inscrit

-Doit pouvoir ajouter un employé a son équipe et définir les jours travaillé ainsi que la pause repas et les vacances

-Sur le compte de la société doit pouvoir voir les informations de ses employés ainsi que leur planning, doit pouvoir aussi retirer un employé.

*Pour les employés :*

-Doit pouvoir compléter sa fiche (adresse employer, complément adresse employer, code postal employer, ville employer, téléphone employer, profession employer, image employer, date de création employer)

-Ne peux être rechercher que si la partie précédente est faite

-Doit pouvoir voir ses rdv du jour et de la semaine en fonction de chaque heure et de voir la fiche du patient et animal en fonction de ses rdv (consultation uniquement), doit pouvoir prendre un rdv en « live », possibilité d’annuler un rdv, possibilité, valider ou annuler le rdv.

-Doit pouvoir inscrire pour les patients les médicaments et la posologie

*Pour les particuliers :*

-Doit pouvoir compléter sa fiche patient (adresse patient, complément adresse patient, code postal patient, ville patient, téléphone patient, image, date de création, date fin patient)

-Doit pouvoir inscrire un ou plusieurs animal (prénom, date de naissance, type, race)

-Rechercher tous les professionnels suivant leurs disponibilités dans une zone géographique de 10km

-Prendre un ou plusieurs rdv

-Pouvoir consulter ses rdv futur comme passé

-Si possible envoyé un mail 48h avant le rdv

*CONCEPTION VISUEL :*

*Pour le Header et le Footer*

Un dégrader de white au purple (#800080) 

*Dans le body*

Les mots importants seront en purple.

*Couleur Générale d’écriture :*

Black

*Icones :*

Purple sauf dans le header et le footer

*Footer icone :*

Facebook , Twitter(X), instagram,linkedin

**2.3.1Description Fonctionnelle des besoins :**

Dans un premier temps il faut :

1. **Les sociétés :**

|  |  |
| --- | --- |
| **Fonction : création fiche société** | |
| Objectif | La société devra remplir les informations préalables pour s'inscrire. |
| Description | Via la page d'accueil, le professionnel doit pouvoir avoir accès au lien d'inscription. |
| Contrainte | L’inscription sera soumise à une validation suite à la vérification des informations fournies par le professionnel, en fonction du SIRET, de l'adresse, nom du dirigeant, etc., et devra attendre la validation par nos soins pour accéder à son compte et créer sa fiche personnelle. Ainsi que définir s’il est toiletteur ou vétérinaire |
| Niveau de priorité | Priorité : Essentiel |

|  |  |
| --- | --- |
| **Fonction :fiche directeur** | |
| Objectif | Une fois créé, le titulaire du compte devra avoir les droits pour gérer les fiches du personnel et pouvoir voir les rendez-vous de tout le personnel. |
| Description | Faire une page de descriptif de toutes les informations nécessaires de chaque employé, de pouvoir modifier, supprimer, créer, et supprimer des fiches personnelles. |
| Contrainte | Il faudra faire un system de droits utilisateurs suivant le niveau de responsabilité |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : création fiche du personnel** | |
| Objectif | Créer une fiche pour chaque employer |
| Description | Par l'accès à la page professionnel et en fonction des droits utilisateurs, il faudra afficher ou non certaines informations, ainsi qu'il puisse modifier certaines informations de sa fiche comme le téléphone, l'email, ses horaires de travail et l’amplitude horaire de ses rdv . |
| Contrainte | Gérer les droits utilisateurs pour l'affichage et la gestion des données. |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : Le personnel accède à ses rdv** | |
| Objectif | L’employé doit pouvoir gérer son travail sur cette page. |
| Description | Une fois connecté, il doit pouvoir accéder aux informations suivantes :  Rendez-vous du jour (heure, nom de la personne) pris ou libre, en cliquant sur le rdv Afficher les informations du patient (nom, prénom, téléphone, statistiques de % de rendez-vous validés).  Possibilité pour le professionnel d'annuler le rendez-vous.  Possibilité d'ajouter un rendez-vous sur un créneau libre.  Confirmer la présence ou l'absence du patient. |
| Contrainte | Sur l’annulation d'un rendez-vous, il faudra demander une confirmation supplémentaire. De plus, il ne faut pas permettre au professionnel de modifier les informations du client et de gérer la prise de nouveaux rendez-vous dans ce contexte. |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : Fiche feuille de soins** | |
| Objectif | Le professionnel pourra saisir les informations nécessaires pour le suivi. |
| Description | Mettre les soins pratiqués, les médicaments, les posologies, les traitements, les allergies, les recommandations, etc. Pour les toiletteurs, les shampooings, la coupe, le parfum. |
| Contrainte | Aucune obligation de le faire pour le professionnel mais il faut que ce soit fait sur le bon patient donc avec une forte sécurité. |
| Niveau de priorité | Priorité Souhaitable |

|  |  |
| --- | --- |
| **Fonction : Fiche statistique** | |
| Objectif | Permettre au professionnel de faire un suivi de ses rendez-vous. |
| Description | Faire un suivi du nombre de rendez-vous journaliers, hebdomadaires, mensuels et annuels, ainsi que voir le nombre de rendez-vous annulés et réalisés, en incluant le ratio des deux. |
| Contrainte | Le gérer en fonction de la fiche de l'employé et permettre à la fiche du directeur d'y accéder. |
| Niveau de priorité | Priorité Souhaitable |

1. **Particulier :**

|  |  |
| --- | --- |
| **Fonction : création fiche client** | |
| Objectif | Créer une fiche client |
| Description | Le lien sera dans la navbar et dans le footer pour la connexion et l'inscription. Récupérer les informations suivantes : nom, prénom, adresse, ville, code postal, téléphone, pouvoir ajouter plusieurs animal |
| Contrainte | Bien respecter le RGPD avec l'acceptation des mentions légales et du RGPD. Se servir d'une API d'adresse pour un pré-remplissage des champs adresse, ville et code postal. |
| Niveau de priorité | Priorité Essentiel |
| **Fonction : Enregistrement Animal client** | |
| Objectif | Lister le ou les animaux du client |
| Description | Un client peut avoir un ou plusieurs animaux de compagnie de différentes races. |
| Contrainte | Lors de la prise de rendez-vous il faudra pouvoir choisir l’animal pour lequel le client prendra le rdv il faudra inscrire la race le no la date de naissance |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : session de connexion** | |
| Objectif | Permet d’accéder à ses informations |
| Description | Permet d'accéder aux informations du compte, de pouvoir prendre un rendez-vous en ligne, de voir les rendez-vous pris et les rendez-vous passés, permet aussi d'annuler un rendez-vous et de rechercher un professionnel. Permet également de récupérer le mot de passe en cas d'oubli. |
| Contrainte | Toujours garder la session ouverte et pouvoir se déconnecter à tout moment, gestion des cookies |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : modification fiche client** | |
| Objectif | Page pour modifier supprimer la fiche client |
| Description | Récupérer toutes les information client pour pouvoir la modifier ou supprimer le compte ou ajouter un animal |
| Contrainte | Respecter la réglementation sur les données stocker, ajouter une confirmation pour la suppression et la modification lors de la validation |
| Niveau de priorité | Priorité Essentiel |

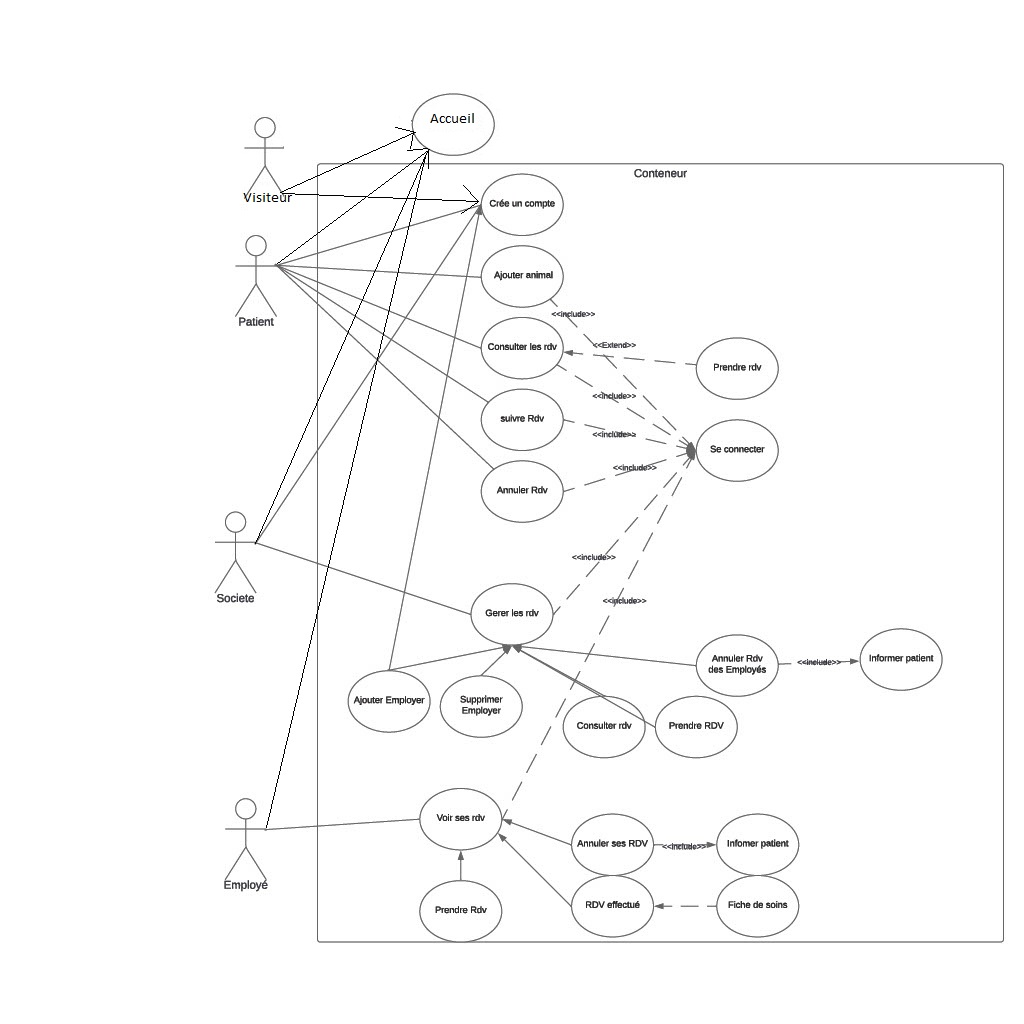
|  |  |
| --- | --- |
| **Fonction : recherche professionnel** | |
| Objectif | Rechercher un professionnel |
| Description | Rechercher un professionnel pour voir ses disponibilité (heure, date) et de pouvoir prendre le rdv en question et affiché le nom, l’adresse, le téléphone |
| Contrainte | Définir un secteur (régional, départemental, ville voisine, dans la ville), au choix de l'utilisateur, rechercher aussi par la date la plus proche en fonction du premier paramètre. |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : voir ses rendez-vous en cours** | |
| Objectif | Voir ses rendez-vous en cours |
| Description | Pouvoir accéder à ses rdv sur la page d’accueil quand l’utilisateur est connecté et fournir un lien aussi |
| Contrainte | Affiché les dates des rendez-vous chez qu’elle professionnel non adresse …, pouvoir annuler un rdv mais impossible de l’annuler si date inférieur à 48h de la prise du rdv |
| Niveau de priorité | Priorité Essentiel |

|  |  |
| --- | --- |
| **Fonction : historique des rdv** | |
| Objectif | Voir ses historiques de rdv passé |
| Description | Pouvoir accéder à ses historiques sur la page d’accueil quand l’utilisateur est connecté et fournir un lien aussi |
| Contrainte | Affiché les dates des rendez-vous chez qu’elle professionnel non adresse … |
| Niveau de priorité | Priorité Souhaitable |

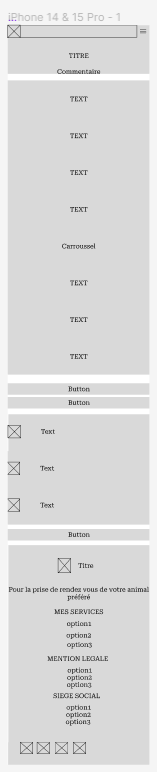
|  |  |
| --- | --- |
| **Fonction : Rappel des rdv** | |
| Objectif | Pour éviter les rdvs non honorés envoyé un mail ou un texto 72h avant |
| Description | Rappel du rdv 72h avant ce qui permettra d’annuler le rdv si le client n’est plus disponible |
| Contrainte | Le client devra pouvoir choisir si il souhaite recevoir une alerte et si oui soit par mail soit par texto ou les deux |
| Niveau de priorité | Priorité Souhaitable |

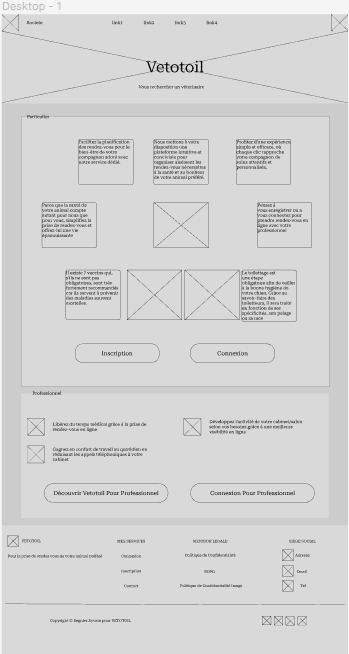
**2.3.2 Uml :**

****

**2.4 Création des WireFrames :**

**Mobile/desktop**





**2.5 Réalisation des maquettes :**

*Version Mobile*

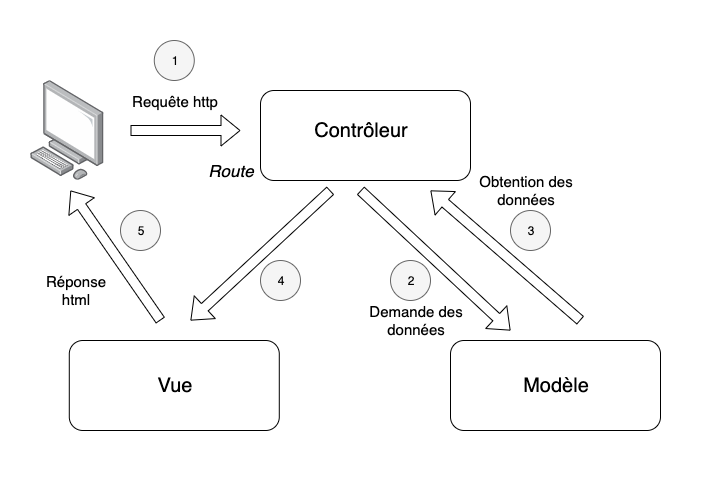
**

*Version PC :*

**

*Architecture du MVC :*

Dans un premier temps voici le fonctionnement du MVC la requête http fait appel à une route qui redirige vers le controller celui-ci suivant l’action à effectuer peux soit retourné une vue qui sera affiché à l’écran de l’utilisateur soit faire une demande de donnée au modèle qui en retour reçoit une réponse qui ensuite le retourne à la vue ;

Ce model est utilisé aussi sur Symfony.

|  |  |  |  |
| --- | --- | --- | --- |
| router | Html ,javascript qui peux aussi avoir du php | class php | class php qui contient les requete sql |

**03 Introduction et structure :**

Pour la partie Front-End, j'ai opté pour l'utilisation de HTML, du framework Bootstrap, de Javascript, de Twig, et d'un peu de CSS, le tout intégré dans le cadre du framework Symfony et de son architecture. J'ai également fait appel à des librairies servant à la fois le Front-End et le Back-End, telles que :

-Le recaptchaV3 de Google fournit par Karser (<https://github.com/karser/KarserRecaptcha3Bundle>)

-VichUploaderBundle pour insérer des photos et les affichés (<https://symfony.com/doc/current/controller/upload_file.html>)

J'ai aussi utilisé l'API ADRESSE de l'État, permettant à l'utilisateur de trouver automatiquement la ville et le code postal en fonction de sa saisie.

Pour obtenir un rendu en mode développement, il est nécessaire de lancer au préalable le serveur de Symfony avec la commande appropriée.

symfony server:start

Puis on accède au site a l url suivante http://localhost:8000

3.1 Voici la structure de la partie Front-End

|  |  |
| --- | --- |
|  | Dans les assets nous retrouvons un dossier js ou tous les scripts sont inséré dedans.  Nous trouvons aussi le dossier styles ou il y a le fichier css |

|  |  |
| --- | --- |
|  | Dans cette partie, située dans le dossier public, j'ai créé un dossier nommé "images" contenant quatre autres dossiers.  Le dossier "img" est destiné à toutes les images du site.  Les trois autres dossiers sont utilisés lorsque les utilisateurs s'inscrivent et peuvent ajouter une photo à leurs profils via VichUploader ; ces dossiers sont organisés en fonction de leurs rôles.  Ensuite, on trouve le dossier "templates", qui contient les vues avec l'extension .html.twig. Cela indique que l'on peut y intégrer du code HTML, et l'utilisation de Twig permet de récupérer des variables pour rendre le site plus dynamique et non statique.  Le fichier **base.html.twig** sert de modèle de référence pour les autres pages du site. C'est dans ce fichier que l'on intègre le script Bootstrap ainsi que l'importmap App, lequel fait référence au fichier **app.js**. Dans ce dernier, le chemin vers le fichier de style **app.css** est déclaré. Cette configuration est automatiquement gérée par Symfony, facilitant ainsi l'intégration de ressources CSS et JavaScript dans l'ensemble du projet |

Voici le fichier base.html.twig

|  |
| --- |
|  |

Dans ce fichier, nous trouvons donc les liens pour accéder à la bibliothèque Bootstrap ainsi qu'aux icônes Bootstrap.

De plus, Symfony génère automatiquement un bloc JavaScript dans lequel se trouve l'importation de **'app'**, permettant ainsi de retrouver l'import du fichier CSS pour qu'il soit actif sur toutes les pages du site.

Ensuite, toutes les pages seront intégrées dans la balise **<body>**. Importer le header et le footer une seule fois permet de les rendre visibles sur l'ensemble des pages du site sans nécessité de réimportation. Concernant le **{% block body %} {% endblock %}**, celui-ci doit être inclus dans toutes les pages pour indiquer au navigateur où le contenu spécifique à chaque page doit être affiché.

Interface web adaptable

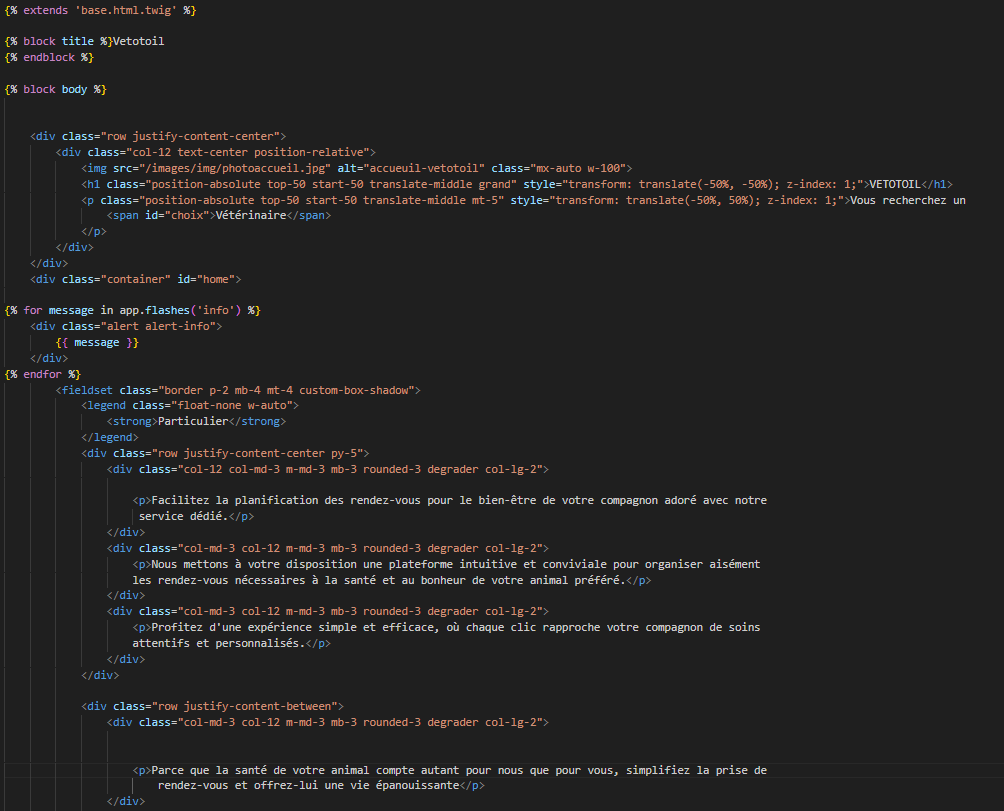
Nous avons vu le fichier principal préalablement et pour faire de notre site un sire responsive nous avons besoin de rajouter cette ligne.

<meta name="viewport" content="width=device-width, initial-scale=1">

Qui permet de dire au navigateur qu’il doit s’adapter à la largeur de l’appareil le viewport définissant les propriétés d’affichages.

Grâce à la bibliothèque Bootstrap, gérer le design responsif devient plus simple, réduisant ainsi le besoin d'écrire abondamment dans le fichier CSS. Bootstrap offre un système de grille flexible, des composants réactifs prédéfinis et des classes utilitaires pour le responsive design, permettant aux développeurs de créer des sites qui s'adaptent élégamment à différentes tailles d'écran avec un effort minimal.

Voici une partie du [home.html.twig](http://home.html.twig)



Dans un premier temps, on étend le fichier **base.html.twig** pour bénéficier de l'affichage prédéfini, ce qui évite de répéter l'inclusion des liens vers Bootstrap, le header, etc., à chaque nouvelle page. Ensuite, on insère le contenu spécifique de la page dans la balise **{% block body %}**. À la fin de ce contenu, il est nécessaire de fermer ce bloc avec **{% endblock %}** pour délimiter où le contenu dynamique doit être injecté dans le template de base. Cette méthode permet de maintenir une structure cohérente et réutilisable sur l'ensemble du site, tout en personnalisant le contenu de chaque page.Dans cette partie de code et avec bootstrap tous se fait au niveau des class.

Les row définissent une ligne et les colonnes sont une division de cette ligne, qui ne peux excéder 12 colonnes et chaque colonne peux encore avoir 12 colonnes.

Cette ligne veut dire qu’il y a une ligne et qu’elle est centré horizontalement dans son conteneur parent :

<div class="row justify-content-center">

Ici j’informe le navigateur que je veux que la colonne est une taille de 12 qui correspond à une colonne de 100% de largeur par rapport au paramètre précédent (s’il y a une colonne de 1 alors il prendra 100% de la colonne et non de la page).

Le text-center position-relative permet au contenu de cette colonne d’être centre horizontalement et il est positionné de façon relative par rapport à sa position normale dans le flux du document, je peux donc ajuster son positionnement à ma guise.

<div class="col-12 text-center position-relative">

<img src="/images/img/photoaccueil.jpg" alt="accueil-vetotoil" class="mx-auto w-100">

Ici j’insère l’image avec la balise img, le paramètre « alt » a plusieurs utilités qui sont, en cas d’erreur de chargement de l’image c’est l’annotation qui est indiqué qui sera vu à l’écran, il permet aussi aux personnes ayant des dispositifs de vue comme des lecteur d’écran de leur signifié ce que c’est et le dernier point est qu’il sert aussi pour les moteurs de recherche pour indexer les images (SEO).

Le src permet d’indiqué ou est situé le chemin de l’image et là class bootstrap mx auto défini qu’il faut centrer horizontalement l’image et le w-100 d’utiliser 100% de la largeur du conteneur.

La balise <h1> est la balise utilisée généralement pour les titres c’est l’une des balises les plus importantes pour le SEO.

3.2 Le responsive sous bootstrap

Bootstrap permet de définir dans ses classes les tailles d’écran comme ceci

<div class="col-12 col-md-3 m-md-3 mb-3 rounded-3 degrader col-lg-2">

Ici on définit que la taille initiale est de de 12 colonnes donc 100% de l’écran, mais dès que l’on passe sur des écrans de taille moyenne la taille est réduite à 3 colonnes donc ¼ de celui-ci le m-md-3 permet de faire un margin de 3 sur tous les coté et le mb-3 rajoute une marge en bas ensuite le col-lg-2 indique que pour les écrans larges cela occupera 2 colonnes sur 12.

Comme on peut le voir bootstrap permet de gérer beaucoup de chose mais nous devons quand même passer par du css lorsque ce que l’on demande sort du type bootstrap par exemple pour faire un dégradé avec mes propres couleurs j’ai dû faire du css pour le header et le footer

.navbar,.footer {

    background: linear-gradient(white, purple);

}

Tous comme pour faire un hover sur un bouton sur les taille d’écran supérieur a 1200px j ai dû le faire en css en utilisant les média querie

<button type="button" class="btn btn-custom rounded-pill">Connexion</button>

@media only screen and (min-width: 1200px) {

    .btn-custom:hover {

        background-color:pink !important;

        color: purple !important;

    }

    .grand{

        font-size: 500%;

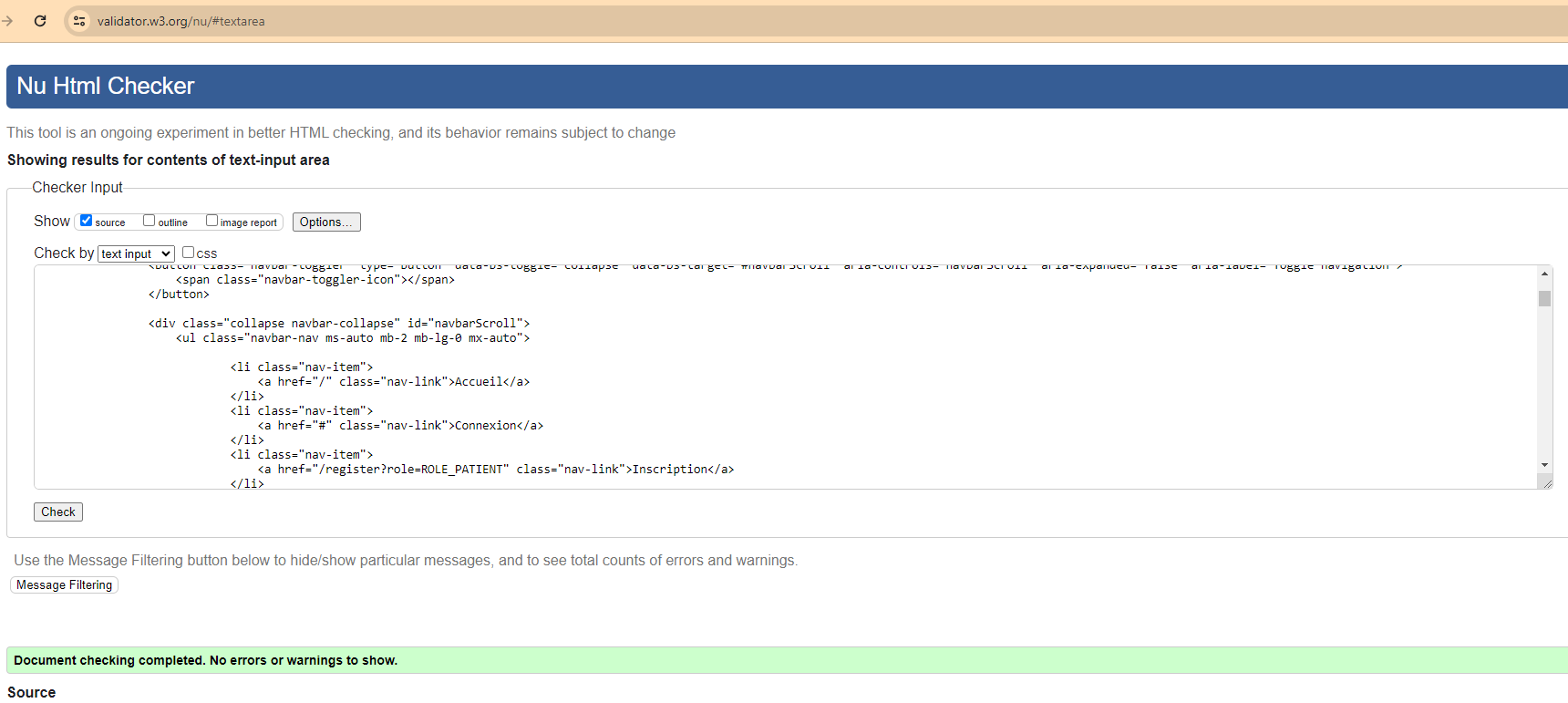
    }

}

Donc pour établir ce site j’ai donc commencé par le mobile first puis adapter mes class bootstrap en fonction des tailles d’écran.

3.3 Validation W3C Validator

J’ai passé mon code de la page d’accueil sur le validateur W3 Validator, étant donnée qu’il y a du twig dedans j’ai dû copier la balise « Body » dans la console pour le tester le document n’a pas d’erreur ou d’alerte ce qui est pour le SEO l’un des critères



3.4 Accessibilité

Dans tous les projets et surtout pour le SEO l’accessibilité est importante j’ai donc utilisé Wave wab accessibility evaluation tool pour vérifier les erreurs



3.5 Partie dynamique :

Ayant la partie statique j’ai donc ajouter du JavaScript pour le rendre plus dynamique sur certaine fonctionnalité, dans un premier temps sur la saisie du mot de passe j’impose un nombre minimum de chiffre ET lettre, une majuscule et un caractère spécial dès que l’utilisateur commence à taper une « ul » s’affiche avec la liste qui se valide au fur et à mesure, si un champ est manquant la validation ne se fait pas

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | Dans un premier temps je mets un écouteur pour confirmer le chargement du DOM  Ensuite je recherche l’id du submit  Met un écouteur dessus avec la fonction « click »  Vérifie, même si symfony le gère la concordance entre les 2 mots de passe  Vérifie la longueur du mot de passe avec. length  Vérifie s’il y a une majuscule  S’il y a un chiffre minimum  Si un caractère spécial est bien présent  Et pour finir l’alert dialog si un champs n’a pas été respecté en utilisant la variable déclaré errors en tableau et qui est push (insérer) au fur et à mesure des erreur détecté |

|  |  |
| --- | --- |
|  | Dans cette suite de code je refais les vérifications en direct pour afficher ou masquer les champs qui ne respecte pas la chartre des mots de passe en rendant visible l’image ou en la rendant invisible mais aussi en rendant invisible l’ « ul » si le mot de passe est vide |

Pour la partie société J’ai fait appel à l’API SIRENE proposé par le gouvernement pour permettre une facilité de saisie en fonction du numéro de siret de la société le tout fais en javascript avec la method fetch

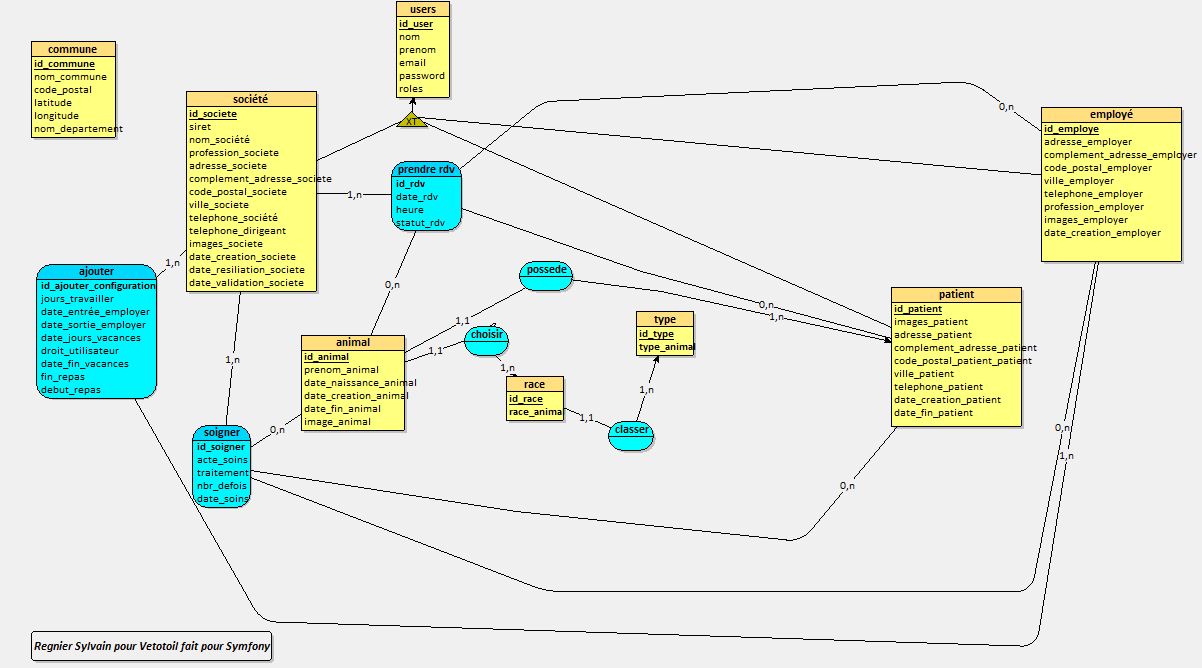
|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | Définit l’url de l’api  Clef d’accès de test de l’api(version de test)  J’intercepte les lettres qui sont appuyé pour autoriser uniquement les chiffres et certaine touche  Je renseigne le lien de l’url enregistré + la valeur du champs input  Method fetch ou je passe les paramètres ou je passe les paramètres obligatoires dans le header demandé par l api  Je récupère en json le résultat  Je récupère les informations nécessaires  J’insère les information récupéré dans l input |

**04 Back-End du Projet**

**04.0.1 MCD:**

J’ai donc élaboré un mcd pour pouvoir construire ensuite le mld qui sera ma référence pour ma base de donnée et mes tables. Au départ je les ai faits à la main sur une feuille puis j’ai utilisé looping pour un meilleur rendu.



Pour élaborer le MCD je suis partie d’une entité user qui a un héritage sur l’entité société, patient et employer.

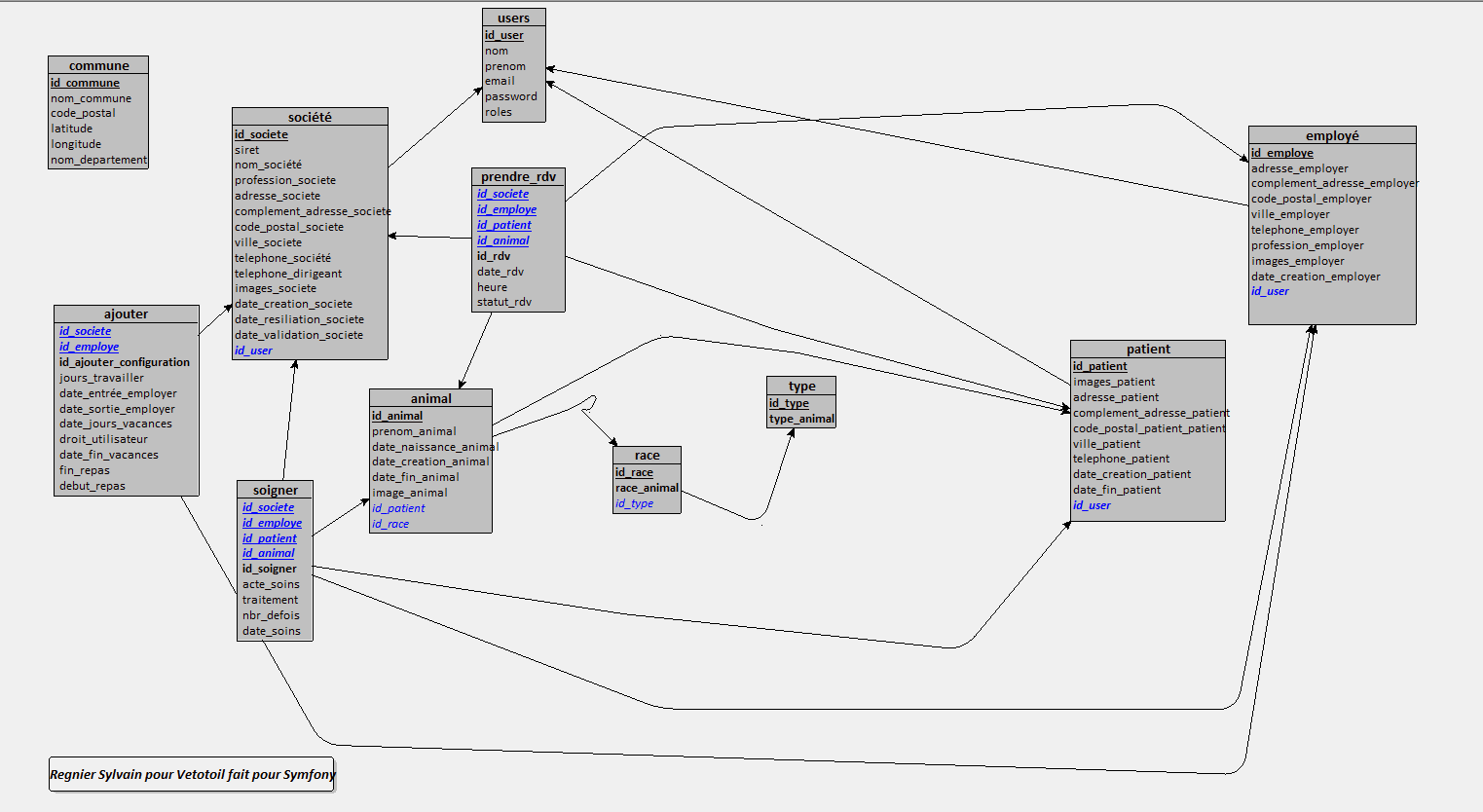
De la partie société une association a été créé pour ajouter des employés. De l’entité société à ajouter j’ai mis une cardinalité de 1, N car une société peut avoir un ou plusieurs employer et d’employé à ajouter la cardinalité est de 1, N car un employé peut être affecté à une ou plusieurs sociétés. Sur le MLD cette association deviendra une table car nous avons d’un côté N et de l’autre côté N.

Entre l’entité animal et race l’association choisir ne deviendra pas une table au MLD car l’association est d’animal à choisir avec la cardinalité 1,1 ce qui correspond que l’on peut choisir 1 seule race pour un animal. De race à choisir la relation est du 1, N car une race peut être choisit pour 1 ou plusieurs animaux et donc une clef étrangère dans l’entité animal suffira car c’est une relation directe.

L’association soigner et prendre rdv qui a une cardinalité N N sera modifié pour les mêmes raison dans le MLD

L’association possède et classer sera du même ordre que l’association choisir car nous avons du 1,1 en cardinalité

4.0.2 MLD :

**

Pour le MLD j’ai donc ajouter les clefs étrangères et fait les tables en fonction du mcd ce qui m’a permis de pouvoir commencer le développement du projet

Pour cette partie j’ai utilisé la structure de Symfony , c’est-à-dire un Controller,Entity,Form,Repository a cela j’ai ajouté un dossier function pour mettre des fonctions et pour insérer des données personnel dans la base de de donnée(bdd) j’ai utilisé la fonction de symfony, dedans j’ai mis des données statique pour les races d’animaux , les types ainsi que tous les renseignements pour les communes de France(info récupéré sur le site du gouvernement) pour faire un système de zonage.

composer require --dev orm-fixtures

4.1 Squelette symfony

Voici le squelette de l’architecture de symfony pour la partie Back-End

|  |  |
| --- | --- |
|  | Dans le dossier src  Nous avons la partie controller qui réceptionne les requêtes des utilisateurs, traite la logique, permet de manipuler les données et de sélectionner et préparer la vue, il sert de lien entre l’utilisateur, les données de l’application et l’interface utilisateur  DataFixture pour intégrer les données fixe ou fake dans les tables.  La partie entity permet de manipuler les données dans la base de donnée avec les getter et setter comme des objets. Une entité correspond aux colonnes d’une table, dedans nous avons les annotations lier au colonnes de la table le type les relations  Le Form permet de crée, traiter et valider des formulaires plus rapidement, ils peuvent être réutilisable sur différente vues  Le dossier function que j’ai créé pour mettre certaine fonction de traitement réutilisable  Le Repository sert de relation entre l’application et la base de donnée c’est dedans que nous avons les requêtes prédéfinies par Symfony mais l’on peut aussi ajouter des requêtes complexe que Symfony ne gère pas |

Il y a aussi les dossiers indispensables à Symfony qui sont créé automatiquement comme le dossier vendor.

4.2 Base de Donnée

*4.2.2 sous symfony*

Sous symfony pour crée la base de donnée nous avons besoin de Mysql utilisé avec phpMyAdmin pour la visualisation, j’ai utilisé celui de XAMPP.

Pour configurer crée la base de donnée il faut renseigner dans le fichier. env au niveau de la partie

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

Et j’ai rajouté la ligne pour dire a Symfony comment se connecté à la base de donnée

 DATABASE\_URL="mysql://root:@127.0.0.1:3306/vetotoil\_symfony?charset=utf8"

Mysql est gestionnaire de base de donnée utilisé, root et le nom d’utilisateur pour me connecter à la base de donnée, root est par défaut mais il peut être différent si lors de la configuration a la base de donnée nous avons modifier le paramètre utilisateur. Ensuite nous avons l’adresse ip pour mon cas le serveur étant en local 127.0.0.1 est l’adresse local, ensuite nous avons le port 3306 qui est le port par défaut puis le nom de la base de donnée et le jeu de caractère qui permet une compatibilité entre les différentes langues et caractère.

Une fois cela fait nous pouvons lancer la commande pour créer la bdd avec cette ligne de commande

php bin/console doctrine:database:create

Ceci crée la base de donnée mais sans table à l’intérieur

*4.2.3Sous le MVC*

La procédure est différente dans un premier temps il faut crée la table directement dans

phpmyadmin puis créer manuellement les tables et les colonnes

Une fois cette étape faite nous devons faire un constructeur dans le model du MVC comme ceci



La fonction new PDO() permet de se connecter a la base de donnée en lui informant que nous sommes en local avec localhost ensuite le dbname pour se connecté a la base de donnée vetotoil en paramètre le nom d’utilisateur root et en dernier champs le mot de passe à vide (car j’en ai pas mis)

Voici les bases de données suivant les types de méthodes utilisé

|  |  |
| --- | --- |
| Symfony | MVC |
|  |  |

4.3 Création *des entity et des controllers:*

*4.3.1 sous symfony*

Sous Symfony pour crée les tables nous passons par Doctrine qui permet de crée des tables ses relations, ses attributs mais aussi de manipuler les données car il facilite le CRUD et de faire des requêtes DQL qui permette de faire des requêtes en utilisant les concepts orienté objet

Pour crée une table il faut dans un premier temps crée une entity avec cette commande

php bin/console make:entity

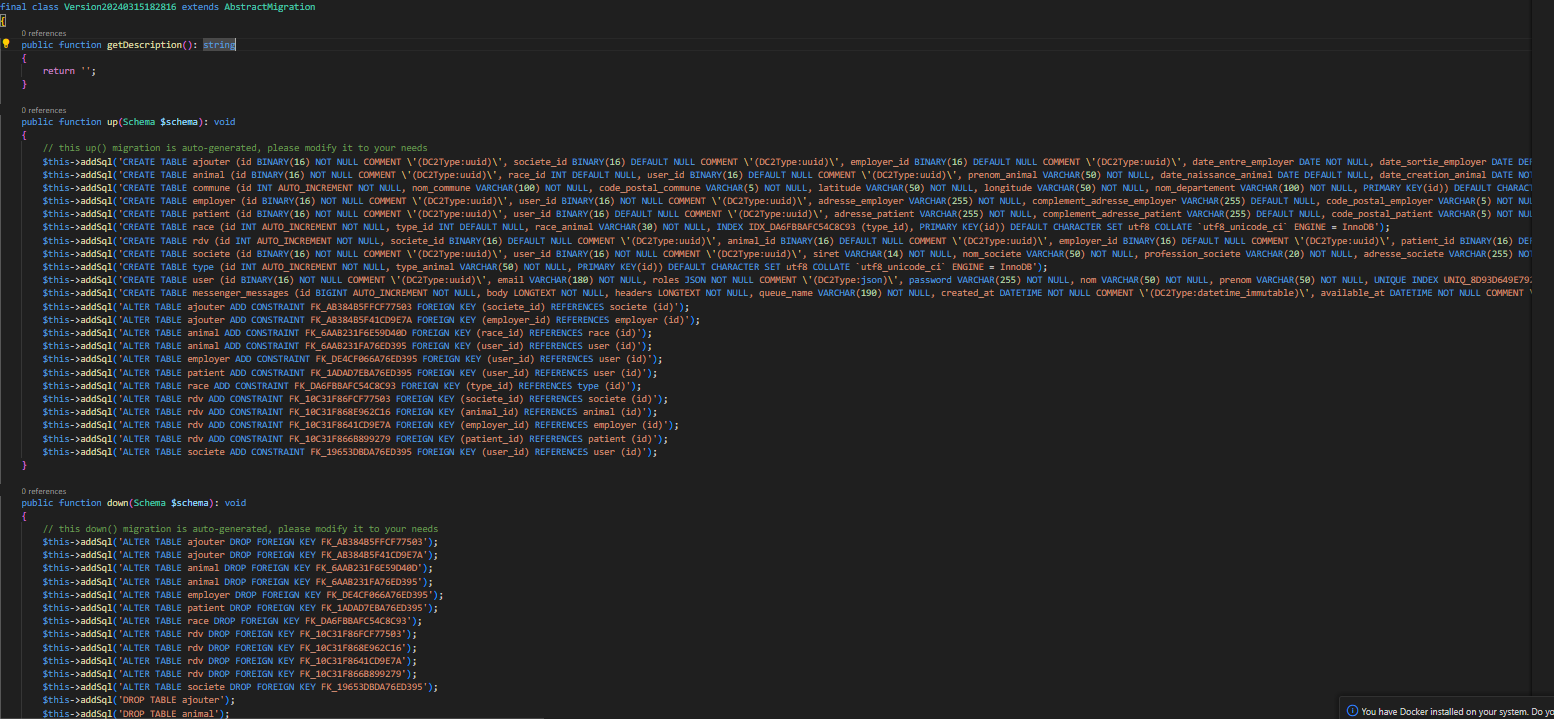
Grace à cette commande nous pouvons crée une table avec des getters ou setters suivant les renseignements et crée aussi le repository de son entity voici un exemple d’entity :

|  |  |
| --- | --- |
|  | Dans un premier temps il nous demande ne nom de l’entity qui sera le nom de notre table en nous donnant un exemple de convention avec la majuscule  Il nous demande le nom du champ  Puis le type de champs  Vu que c’est un string il nous demande la longueur  Et pour finir il nous demande si le champ peux etre Null ou non.  L’id étant créé automatiquement il n’est judicieux d’en crée un |

Une fois crée nous pouvons faire un

php bin/console make:migration

qui va générer un fichier de migration ci-joint un exemple de fichier de migration



Dans celui-ci on peut voir qu’il a créé des commandes SQL avec CREATE TABLE il met le nom de l’entity que j’ai créé qui sera utilisé pour la table dans phpmyadmin avec tous les champs renseigner et ses descriptifs qui sont dans le dossier Src/Entity/ il reprend toutes les entity les relations les clef … pour pouvoir insérer les données dans la bdd.

Une fois le fichier généré il faut faire un migrate pour insérer les tables et les champs dans la base de donnée avec cette commande

php bin/console doctrine:migrations:migrate

Ci-joint mon entity user crée avec la commande

php bin/console make:user

|  |  |
| --- | --- |
|  | Symfony utilise le « namespacing » pour importer des class de son squelette ainsi que des classes que j’ai importées.  Il renseigne aussi le repository affecté pour son entité  Ici on informe que l’email est un champ unique  Il définit le type, le nom des champs et s’il peut être Null ou non  Comme l’on peut le voir j’ai importé la librairie Uuid pour me servir de l’Uuid V7 avec cette commande  composer require symfony/uid  Une fois importé je l’ai intégré a quasiment toutes les tables sensibles pour que les id soit généré de façon aléatoire et que sur chaque Entity il soit aussi unique  Et symfony gère automatiquement le hashage du mot de passe mais du code dans le controlleur est nécessaire |

|  |  |
| --- | --- |
|  | Il est aussi précisé le type de relation entre les entity dans ce cas l’entity user car un user peux avoir plusieurs animaux car c’est du OneToMany, mais un animal ne peut avoir plusieurs propriétaires  Sur cette partie nous avons les getters (lire)  Ici nous avons les setters (modifier,enregistrer)  Il crée aussi un champs rôles (droit pour les utilisateurs du site) automatiquement  Ensuite il faut définir les rôles dans config/packages/security.yaml     access\_control:           - { path: ^/patient, roles: ROLE\_PATIENT }           - { path: ^/societe, roles: ROLE\_SOCIETE }           - { path: ^/employer, roles: ROLE\_EMPLOYER }           - { path: ^/administrateur, roles: ROLE\_ADMINISTRATEUR } |

Maintenant pour pouvoir utiliser cette entity j’ai besoin de crée un controlleur avec la commande

php bin/console make:controller

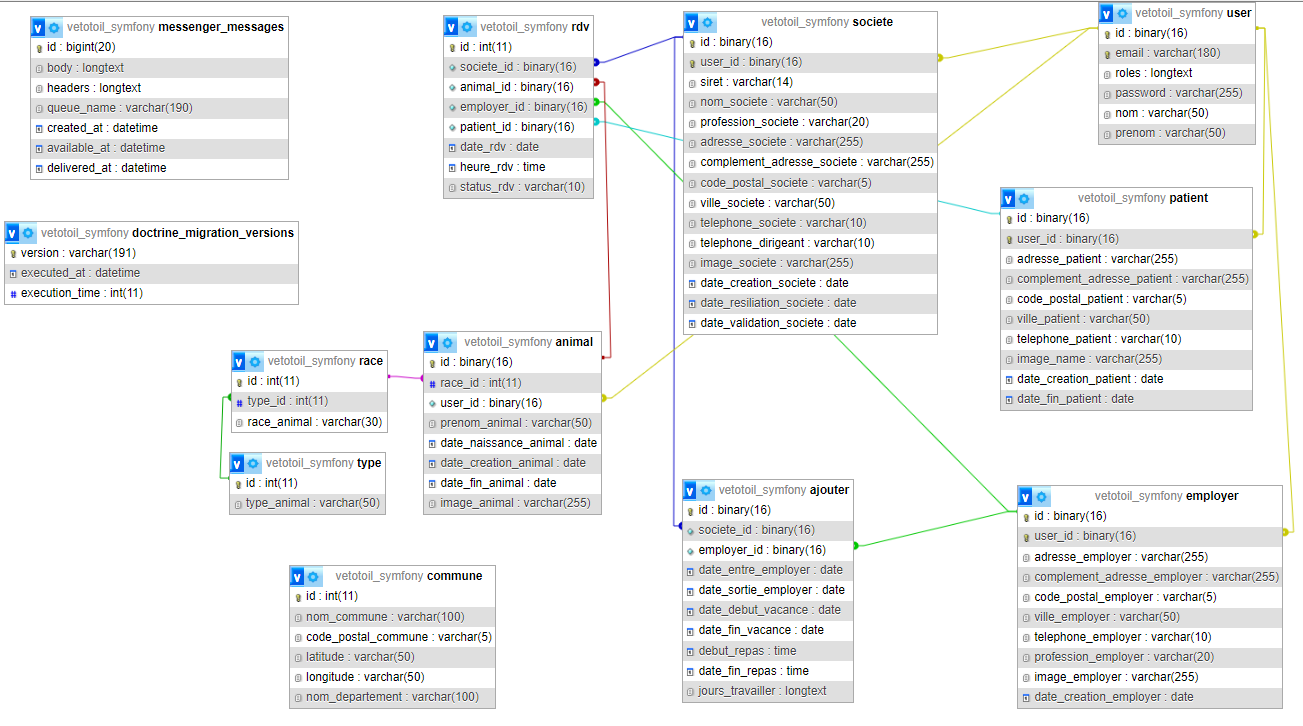
|  |  |
| --- | --- |
|  | Encore une fois symfony nous aide sur la convention il faut mettre chaque première lettre en majuscule et finir avec Controller même si sur la version 7 que j’utilise symfony le rajoute de lui-même.  Renseigne le nom  Puis symfony crée le controlleur mais aussi un dossier du nom du Controller avec la vue index.html.twig |

Voici le controller crée

|  |  |
| --- | --- |
|  | Ici on peut voir que symfony extends la class AnimalController à AbstractController en réalité on hérite de la class crée par symfony pour simplifier le code et réduit le code répétitif  Il crée aussi une route par défaut pour notre controlleur qui nous sera utile pour afficher les vues ou faire des traitements sans vue  Puis dans la fonction il insère directement le render de la vue qu’il a créé ce qui permet un gain de rapidité |

*4.3.1.1Modèle physique de données*

Représentation graphique de la structure physique de ma base de données avec l’intégration des index et des contraintes de clé, C’est la structure finale de nos données stockées vu sur phpmyadmin après avoir fait les relations entre mes entités



*4.3.2 Création des entity et des controllers sous le mvc (model vue controller) :*

Sur le mvc c’est un peu plus complexe car il faut d’abord avoir le fichier index qui sera le gestionnaire des controlleurs ainsi que des models et des autres fichiers.

|  |  |
| --- | --- |
|  | Les require\_once permette de charger qu’une seule fois la page demandée sans avoir à la recharger constamment ici ce sont le header le controlleur qui va gérer le fonctionnement des autres Controller et le model  A chaque nouveau Controller il faut le renseigner dans la variable Controller sinon ce Controller ne sera pas reconnu  On lui précise le Controller par défaut  On commence par vérifier si un paramètre nommé controller est présent dans l'URL en utilisant **isset($\_GET['controller'])**, ce qui assure que le paramètre n'est pas null. Si ce paramètre est présent, le code récupère sa valeur. je vérifie ensuite si cette valeur se trouve dans le tableau **$controllers**, qui contient une liste de noms de contrôleurs valides autorisés par l'application. Si la valeur de **$\_GET['controller']** correspond à un élément dans le tableau **$controllers**, alors la variable **$nom\_controller** est définie avec le nom du contrôleur récupéré de l'URL.  On recrée une nouvelle variable $nom\_classe en concaténant Controlleur\_ car tous les noms des controlleur commence par ceci puis en ajoutant la variable récupérée préalablement pour ensuite accéder au controller  C’est ici que l’on détermine avec cette partie du code quel controller doit être chargé et instancié en fonction de la requête de l'utilisateur ceci est donc le routage |



Ceci est le controller principale qui va gérer tous les autres Controller car ceux-ci seront extends a celui-ci. Ce Controller principale permet de de récupérer l’action qui est renseigner dans le Controller enfants puis d’afficher les view renseigner dans le controller enfant.

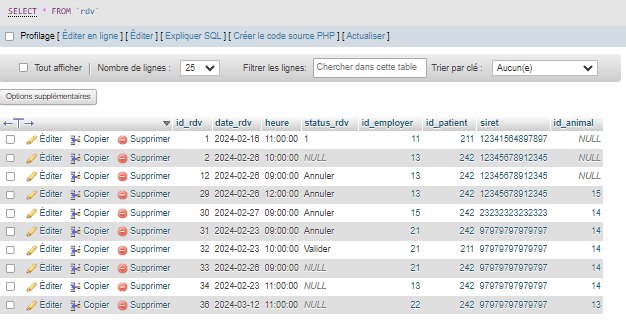
Voici le type de controller qui a pour nom Controller\_employer

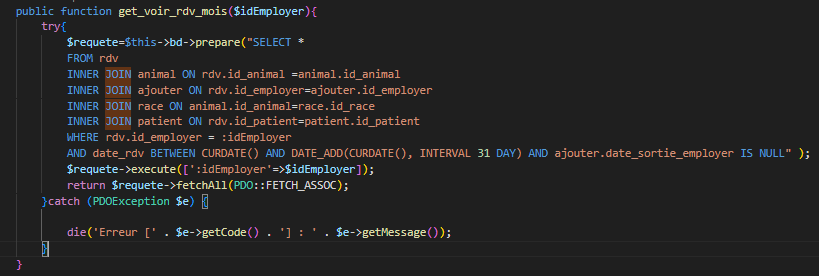
|  |  |
| --- | --- |
|  | On fait appel à la class qui est étendu à Controller  Sur la function action\_home lorsque l’on appel ce Controller avec pour action home on retourne sur la vue home.  Sur la function action\_session\_connect\_employer on fait appel au model grâce Model ::get\_model qui est une instance de la class Model    public static function get\_model()      {          if (is\_null(self::$instance)) {              self::$instance = new Model();          }          return self::$instance;      }  Ensuite je crée une variable $connecteEmployer sous forme de tableau avec pour clef connectEmployer ou j’envoie au model $m à la function get\_connexion\_employer les valeurs du formulaire envoyé par la méthode POST du form .  Puis après le traitement du model je retourne la vue session avec comment variable de traitement $connectEmployer |

Voici le code dans le model

|  |  |
| --- | --- |
|  | Dans cette fonction on commence a récupéré le tableau envoyé précédemment (array $data).  Je récupère l’email de connexion ainsi que le mot de passe qui sont dans mon tableau $\_POST[‘email’]et[‘password’]  Ensuite, je procède à une requête préparée pour rechercher tous les champs de la table employer où la colonne email correspond à l'email saisi. J'utilise un paramètre nommé :email dans ma requête préparée, et je lie la valeur de la variable $email à ce paramètre lors de l'exécution de la requête, ce qui me permet d'effectuer la recherche de manière sécurisée.  J’exécute la requête sous forme de tableau en associant le paramètre à sa variable  Puis je lance le fetch pour retrouver le premier résultat trouvé.  Et pour pouvoir retourné le résultat je vérifie que le hashage du mot de passe correspond bien au mot de passe avec password\_verify  Si le résultat est correct il renvoi au controller le résultat de $requete tandis que si c’est faux il renvoi false  Et bien sûr il y a la gestion des erreurs si la requête c’est mal exécuté |

4.3.2.1 Exemple de requête SQL avec jointure et de la table Rdv



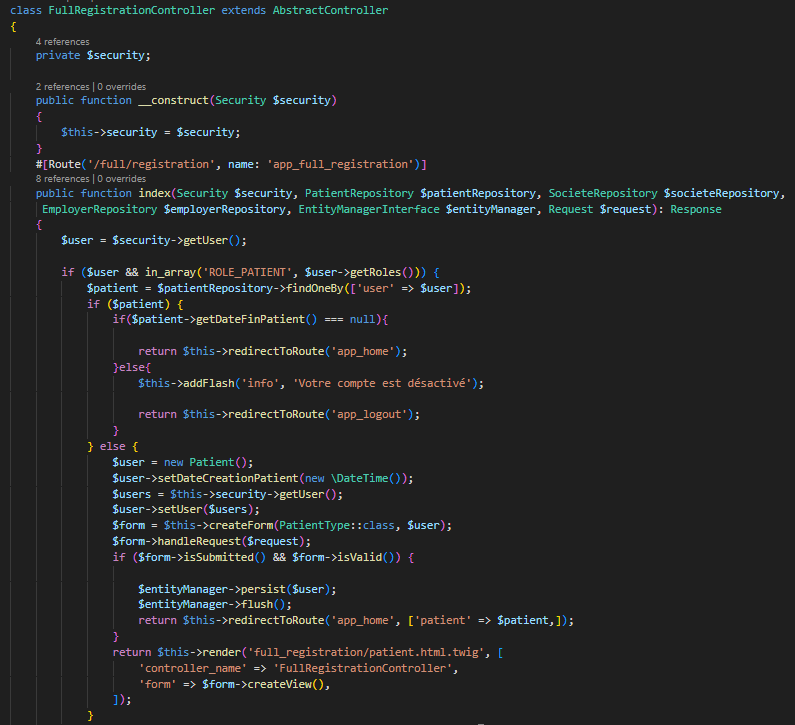


Dans la table rdv j’ai les clef étrangère suivante id\_employer , id \_patient, le Siret de l’entreprise car un employé peux travailler dans plusieurs entreprises(cette requête est destiné pour le gestionnaire de la société) et j’ai aussi besoin de l’id animal car un patient peux avoir plusieurs animaux .

Donc pour avoir un affichage des nom prénom j’ai dû faire une requête ou dans la table rdv je joins la table animal pour retrouver le nom de l’animal mais aussi afin de récupéré l id race de l’animal pour que le professionnel connaisse les informations dessus le prénom, date de naissance de cette table animal j’ai la clef étrangère de la race, je fais donc une nouvelle jointure pour pouvoir afficher la race de celui-ci, dans la table ajouter je recherche en fonction de l’id employé récupéré préalablement ainsi que la jointure sur la table patient pour récupéré les information de la table patient le tout sur la condition de recherche que rdv.id\_employer corresponde à l idEmployer récupéré précédemment et dans cette requête pour j’ai définis un espace-temps d’un mois à la date du jours donc je fais un Between pour filtre les dates entre le jours « j » et un intervalle de 31 jours le tout agrémenté d’un condition supplémentaire que l’employé ne soit pas sortie de l’entreprise car ce champs est initialement « null » mais dès qu’il quitte une société le champs prend la date de sortie validé par l’employeur.

4.3.3 Symfony :

Grace au DQL les requêtes sont géré plus facilement et donc la lecture du code est simplifiée



Au départ je crée un constructeur qui fait appel a Security bundle de symfony ,cela me permet de récupérer les informations de l’user connecté

Cela me permet de pouvoir mettre des conditions en fonction de ma variable $user ,sur cet exemple je recherche le role de l’utilisateur et donc si il est patient on continue

Ensuite je crée une variable patient ou je fait appel au repository de patient pour pouvoir effectuer une requete DQL avec le findByOne car dans cette partie il ne peux y avoir qu’un seul utilisateur comme vu precedemment par rapport a l adresse email d inscription .

Puis je vérifie que patient n’est pas nul puis que son compte ne soit pas désactivé avec

if($patient->getDateFinPatient() === null){

ensuite il est redirigé soit vers l accueil avec

 return $this->redirectToRoute('app\_home');

sinon il aura un message sur la page d accueil avec addFlash que je récupère dans le twig

 $this->addFlash('info', 'Votre compte est désactivé');

                    return $this->redirectToRoute('app\_logout');

et ne pourra se connecté.

Si le patient n’existe pas c’est qu il vient de s’enregistrer a partir de ce moment je fait appel a l’objet Patient() avec cette fonction

$user = new Patient();

J’envoie la date du jours a $user

  $user->setDateCreationPatient(new \DateTime());

Ses deux lignes me permettent de passer l id user

Et de l envoye à $user

 $users = $this->security->getUser();

                $user->setUser($users);

Ensuite je fais appel au formulaire crée par rapport a l entity qui a pour nom PatientType

 $form = $this->createForm(PatientType::class, $user);

                $form->handleRequest($request);

Pour créer le formulaire dans la console il faut faire

composer require symfony/form

cela crée un formulaire avec les champs de la base de donnée a remplir

Je vérifie si le formulaire est envoyé et qu’il est valide, on appel l’ EntityManagerInterface qui permet d interagir avec les entity pour gerer le CRUD dans ce cas présent c’est pour insérer des données dans la base.

Ensuite on persiste(enregistre) les données de la variable user qui a été envoyé par le formulaire avec $entitymanager->persist($user)

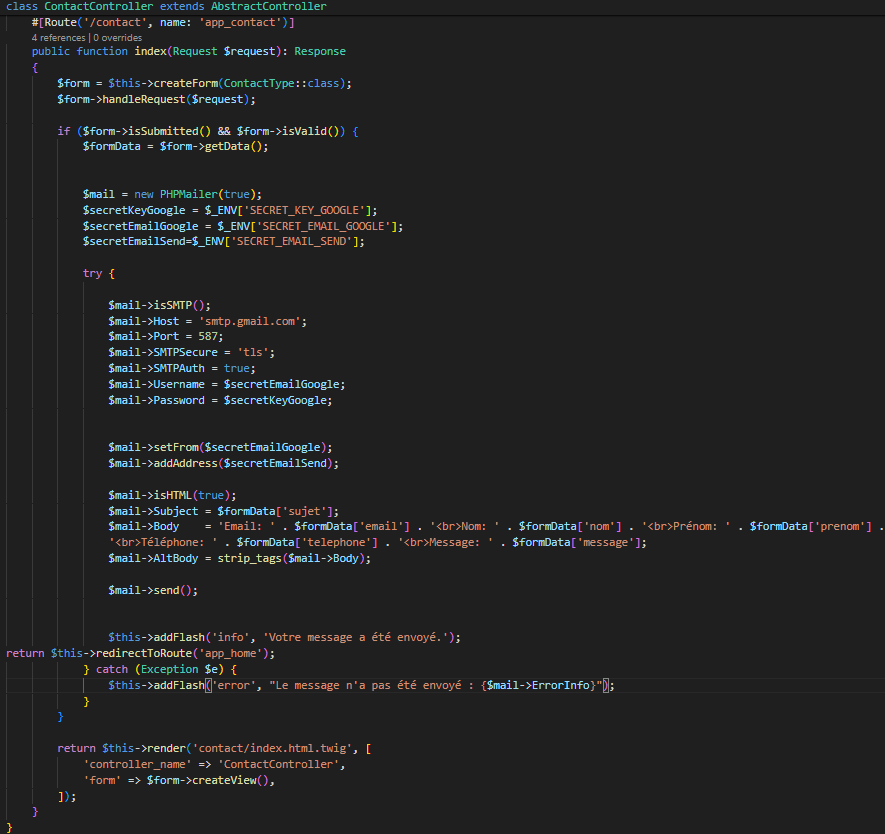
Pour le flush c’est l exécution du persiste c’est a ce moment que les donnée sont rediriger

Et pour finir je retourne l utilisateur sur la page d accueil en envoyant le parametre « patient »

4.4 Ajout de la bibliothèque PhpMailer

Pour le formulaire de contact en back j’ai utilisé phpMailer qui permet d’envoyer des mails ,j’ai donc importé le composant via la console avec cette fonction

composer require phpmailer/phpmailer



Dans un premier temps je crée le formulaire et j’appel ce formulaire

Puis s’il est envoyé et valid on commence le traitement en récupérant les données du formulaire avec $form->getData() qui est mis dans la variable $formData.

On fait appel a l objet PHPMailer puis je récupère les mes clef secrete inséré dans le .env de symfony .

Une fois ce traitement effectué étant donnée que j’utilise google j’ai du crée sur mon adresse email de google une clef sécurisé par rapport a mon mot de passe.Etant donnée que c’est pour l envoie je configure le smt de google le port utilisé le mode de sécurisation pour le chiffrement du mail (tsl ou ssl) et active l authentification du smtp, ensuite PHPmailer demande le nom d utilisateur qui est l adresse email et la clef secrète généré par google.

Puis on defini l’expéditeur et le destinataire pour ce projet l’expéditeur est mon adresse email gmail est le destinataire est mon autre adresse

La ligne

                $mail->isHTML(true);

Indique que le corp du message est de l html .

Puis on reseigne les autre champs qui seront vu dans le mail le sujet que nous récupérons grace a la variable $formData[‘sujet’] et la meme procedure est utilisé pour le reste

La ligne de code suivante

$mail->send();

Envoie le mail .

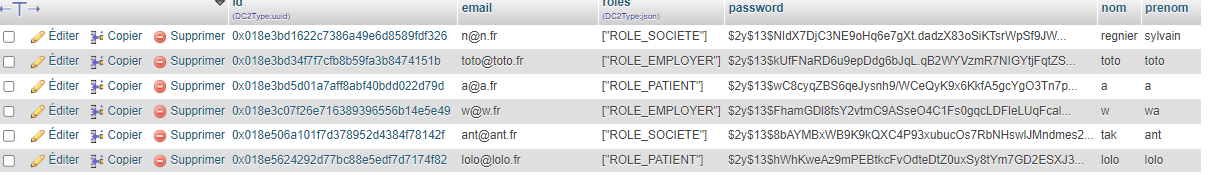
Et pour finir j’ai fait une gestion d’erreur si le message est envoyé il y aura un message sur sur la page d’accueil avec la fonction addFlash et si iil y a une erreur il y aura un message d erreur

4.5 La sécurité :

Depuis toujours la sécurité est tres importante dans tous type de programme.

4.5.1 Les mots de passe

Dans un premier temps j’ai sécurisé les mots de passe avec la fonction de symfony intégré



-Les mots de passe sont hashé grâce à la création du make :user qui est une propriété de symfony pour crée des utilisateurs et de façon sécurisé, symfony gère automatiquement ce hashage. Le premier mot de passe est 123456 une fois hashé il ressort sous la forme $2y$13$NIdX7DjC3NE9oHq6e7gXt.dadzX83oSiKTsrWpSf9JWwdN3Ouh/6S

Bien sûr étant donné que symfony peux modifier pour des raisons de sécurité le code de hashache la taille du champ est a 255

4.5.2 La sécurisation des données inséré par l’utilisateur

-Pour éviter les injections SQL j’ai dans un premier temps, sur les donnée « sensible » passez mes données sous forme de POST avec des formulaire plutôt que des GET ce qui m’évite d’avoir des données dans l url .

J’ai importé la class pour générer des UUID comme déjà précisé qui me permet d’avoir des id généré aléatoirement et qui sont unique encore une fois cela évite les injection SQL car lorsque les champs sont numéroté par des chiffres qui se suivent si il y a une faille ,il est assez facile d’accéder à des donnés que l’on ne devrait pas voir.

Ensuite ne pouvant faire confiance à la saisie des utilisateurs j’ai fait deux fonction une qui est spécialement pour les mails et l’autre pour les champs input ou l’utilisateur insère des données

Voici la fonction pour le champ email :

function email\_form($data)

{

    $data = trim($data);

    $data = htmlspecialchars($data);

    return ($data);

}

Voici la fonction pour les autres champs

function validate\_form($data)

{

    $data = trim($data);

    $data = htmlspecialchars($data);

    $data = stripslashes($data);

    $data = strtolower($data);

    return ($data);

}

Dans ses codes le trim retire les espaces avant et à la fin,

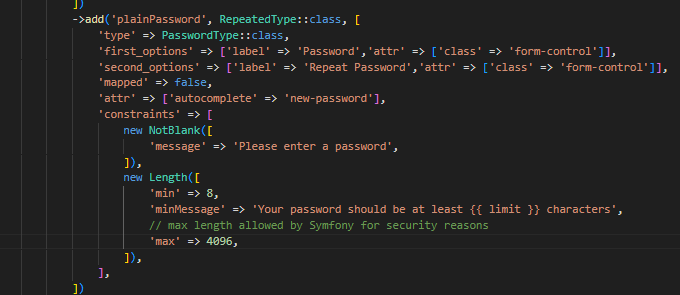
Le htmlspecialchars permet de remplacer certain caractères spéciaux par leur équivalent en entité html exemple le & deviendra &amp cela est utile pour éviter les attaques de type cross-site scripting .puis j’ai mis le stripslashes qui supprime les «  **\**» puis pour éviter les erreur de entre les majuscules et les minuscules je mets tous en minuscule grâce a la fonction strtolower.

Pour mon formulaire d’inscription toujours pour des raisons d’attaque j’ai import é la class karser recaptcha3 qui est le recaptcha de google cela est pour éviter que des robots puisse s’inscrire et éviter les attaques automatisées.

Je suis aussi passé par la sécurité de symfony dans les formulaires

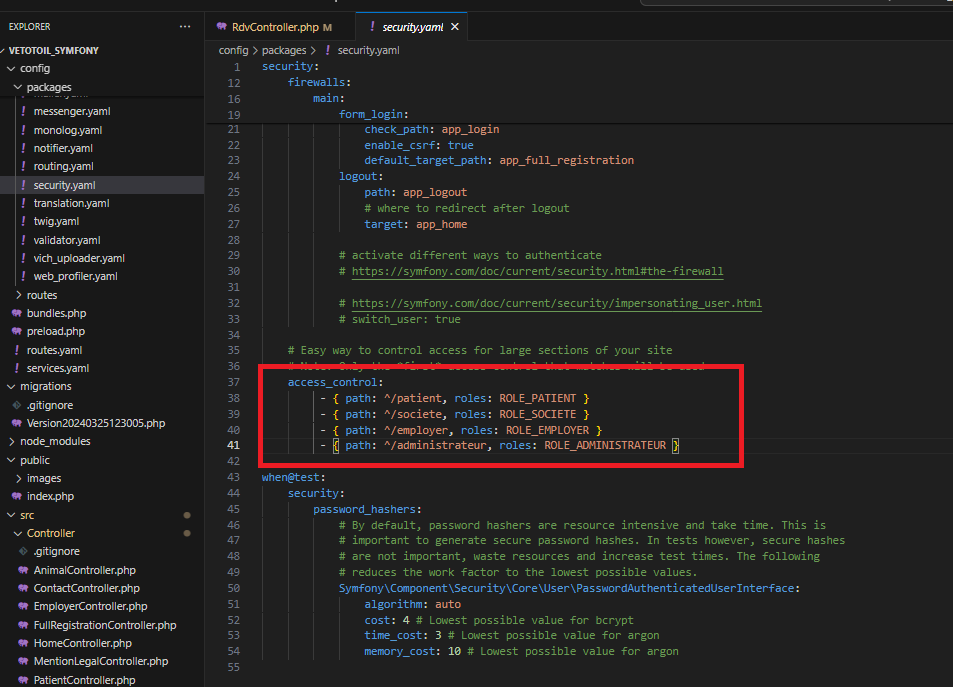
|  |  |
| --- | --- |
|  | Ici l’on peut voir les types de champs ce qui permet d’avoir la sécurité de symfony |

Dans le schéma suivant je rajoute des contraintes au password le minimum de caractère, qu’il ne peut pas être null,



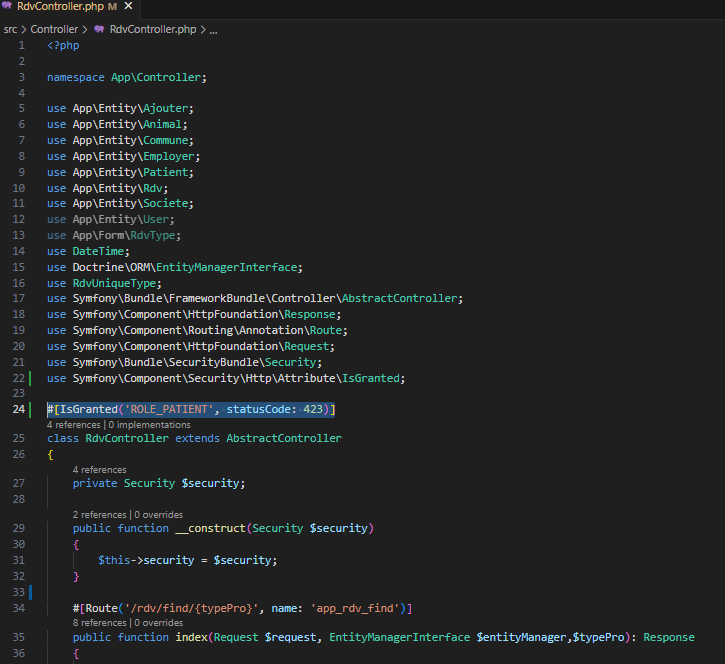
4.5.3Securiser les routes

En ce qui concerne la sécurité des routes dans le fichier security.yaml qui se trouve dans config/packages/security.yaml j’ai au préalable définit dans acces\_control les différent roles pour les différents type d’utilisateur qui sont inséré dans la base de donnée au moment de l’enregistrement.

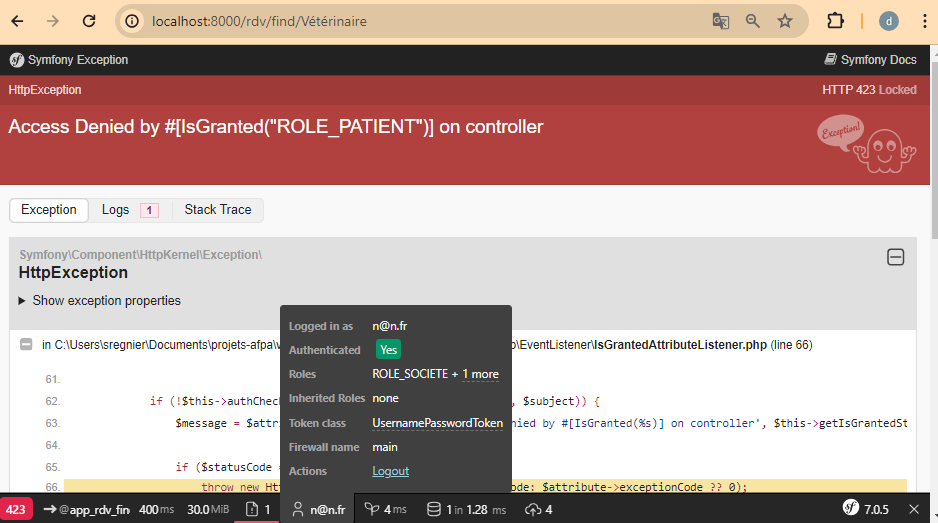


Une fois cette étape faite dans mon controller j’ai importé la class IsGranted de symfony et avant la déclaration de la class j’ai fait avec l annotation IsGranted , la définition de quel role peux accéder a toutes les routes de ce controller pour cet exemple c’est uniquement les patient qui peuvent accéder a celle-ci et j’insere un code d erreur 423 qui sur la partie client -serveur correspond a un verouillage donc le client aura cet erreur

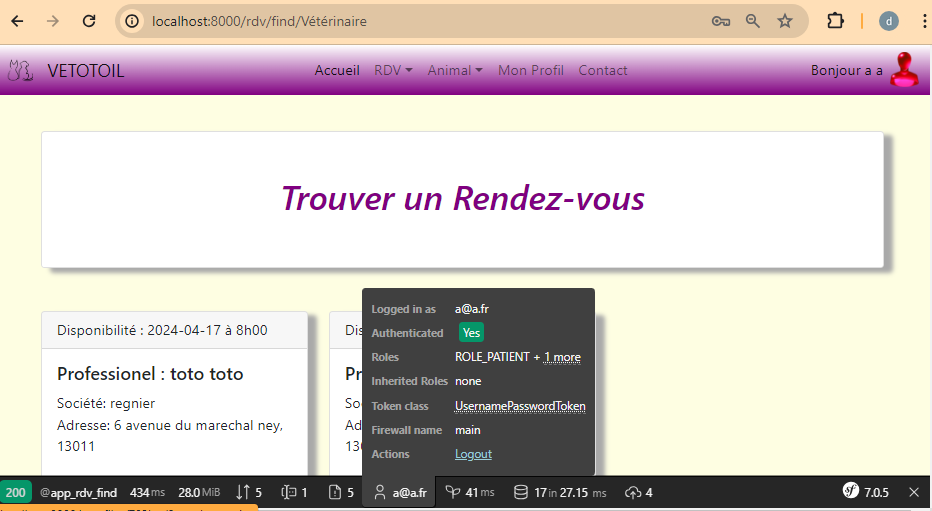
#[IsGranted('ROLE\_PATIENT', statusCode: 423)]



Voici le résultat lorsque qu’une société essaie d’accéder à cette route



Voici avec un utilisateur ayant le role patient



**5-Conclusion :**

Mon travail sur "VETOTOIL" m'a donné l'opportunité d'apprendre les subtilités du développement front-end et back-end, me poussant à trouver des solutions innovantes et efficaces. La partie que j'ai trouvé la plus satisfaisante a été de voir les maquettes statiques se transformer en une interface utilisateur adaptative, ce qui a prouvé à quel point la technologie contribue à fluidifier et enrichir l'expérience utilisateur.

Les différents Framework m’ont permis de gagner du temps sur le développement comme sur la partie front-end avec Bootstrap qui permet de gérer plus facilement le responsive qu’en HTML/CSS uniquement

Pour la partie Back-End la sécurité est un domaine primordial c’est pour cela que j’ai utilisé le Framework Symfony, ainsi que les uuid et le recaptcha de Google pour me prévenir de certaine attaque.

Ayant utilisé une méthode agile Solo, cela m’a obligé à respecter mes délais, de voir aussi les parties plus complexe de code ou je devais faire des recherches et à apprendre à bien gérer mon temps pour chaque activité.

Même si l’application n’est pas totalement terminé, une application se termine-t-elle un jours ?, je suis très satisfait du travail déjà accompli ,des connaissances acquises tout au long de ce projet qui je pense aurais pu être un travail collectif ce qui m’aurait permis de me détacher de certaine tache pour développer de nouvelles fonctionnalité en accord avec le groupe, pour le debug ou chacun développe sa partir et cela m’aurais permis d’avoir du temps en plus pour préparer une application mobile .

**6-Lexique :**

Attaque cross-site scripting(XSS) : est un type de vulnérabilité de sécurité des applications web. Elle permet aux attaquants d'injecter des scripts malveillants dans des contenus qui sont ensuite affichés à d'autres utilisateurs. Une attaque XSS exploite la confiance qu'un utilisateur a pour un site particulier, exécutant du code script (souvent en JavaScript) dans le navigateur de l'utilisateur sans son consentement.

Base de Donnée : C’est une collection organisée de donnée ou dedans nous avons des tables et dans les tables des champs, elle sert à stocker des données et pouvoir y avoir accès

Bootstrap : est un framework front-end qui intègre du css et du javascript prédéfini dans des class qui permet au développeur un gain de temps car il n’a plus besoin de gérer par exemple le positionnement d’un élément en code css et permet d’avoir un design cohérent.

CRUD : Create ,read,Update,Delete , cela signifie les 4 opérations de base dans la manipulation de donnée et pour la gestion de la base de donnée

CSS : permet de faire des styles dans une ou plusieurs page web, cela évite de surcharger le code et de faire une séparation au niveau du code entre un élément de la page html et par exemple la couleur de cet élément.

DBAL : ajoute des fonctionnalités (quelques drivers) mais étend également la notion d'abstraction du simple accès aux données (en PDO) aux bases de données, ainsi Doctrine DBAL permet de manipuler les bases de données en offrant par exemple des fonctions[3](https://fr.wikipedia.org/wiki/Doctrine_(ORM)#cite_note-3) qui listent les tables, les champs, le détails des structures.

Doctrine : Est un projet de mappage objet-relationnel (ORM) pour PHP. Il est conçu pour faciliter l'intégration et la manipulation de données dans des applications PHP, en utilisant des objets pour représenter des données stockées dans une base de données relationnelle

Dom : Le DOM est une interface de programmation pour des documents HTML ou XML qui représente le document (la page web actuelle) sous une forme qui permet aux langages de script comme le JavaScript d’y accéder et d’en manipuler le contenu et les styles.

GITHUB : est une plateforme de développement collaboratif qui permet d’héberger des projets informatiques et de facilité le travail en équipe autour du développement de logiciel

HTML (hyper text markup language : est un langage de balisage pour crée et structurer des pages ou des applications web.Le html permet au navigateur de savoir comment afficher les pages

Injection sql : Est une technique d'attaque informatique utilisée pour exploiter les vulnérabilités dans la gestion des entrées utilisateur d'une application. Elle permet à un attaquant d'injecter des instructions SQL malveillantes à travers des champs d'entrée prévus pour l'utilisateur (comme des formulaires web), dans le but de manipuler ou d'accéder à la base de données sous-jacente de l'application sans autorisation.

JavaScript : est un langage de programmation principalement utilisé pour la partie dynamique du site, il s’exécute sur la partie client

MVC : Le modèle MVC, ou Modèle-Vue-Contrôleur, est un motif d'architecture logicielle qui sépare une application en trois composants principaux : le modèle, la vue, et le contrôleur. Cette séparation aide à gérer la complexité des applications en permettant une séparation des préoccupations, ce qui facilite le développement, les tests, et la maintenance de l'application

ORM : fournit la persistance transparente des objets PHP. C'est l'interface qui permet de faire le lien ou "mapping" entre les objets et les éléments de la base de données (que gère DBAL).

Pdo : c’est une extension de PHP qui fournit une interface uniforme pour accéder à différente base de donnée, sur une programmation orienté object.

PHP : Est un langage qui s’exécute coter serveur qui permet par exemple d’avoir accès aux basse de donnée et de pouvoir les manipuler

SEO : désigne l'ensemble des techniques et stratégies visant à améliorer la visibilité d'un site web dans les pages de résultats des moteurs de recherche

Symfony : Est un framework qui utilise l’architecture MVC qui a pour objectif de simplifier le développement, il est réputé pour sa gestion de la sécurité contre les injection sql par exemple