

TITRE PROFESSIONNEL DWWM



# *TITRE PROFESSIONNEL DWWM*

DEVELOPPEUR WEB ET WEB MOBILE



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
2. 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
3. Introduction et structure
  - 3.1.Structure Front-End
  - 3.2.Responsive
  - 3.3.Validation W3C Validator
  - 3.4.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.Déploiement du site

6.Conclusion

7.Lexique

## 01 | Compétence du référentiel couverte

### *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 des attaques malveillantes comme par exemple en vérifiant chaque donnée insérée par l'utilisateur (injection SQL, injection de script ...), mais aussi en me servant de UuidV7 unique, pour rendre chaque ID aléatoire au lieu d'avoir des chiffres qui se suivent, captchaV3 de Google pour éviter les inscriptions des robots.

-J'ai aussi fait le contrôle sur l'accès aux données et aux fonctionnalités du site via le système des rôles.

-La partie back-end présentée est faite sous Symfony mais une version en MVC a aussi été faite.

-Une partie envoi de mail à l'administrateur est aussi proposée, cela a été fait avec PHPMailer

## 02 Présentation du projet

VETOTOIL est une application qui a pour but de pouvoir prendre des rendez-vous chez le professionnel de votre choix, vétérinaire, toiletteur (liste non exhaustive) sans avoir à rechercher le professionnel sur internet puis de le contacter ou par moment il n'y a pas de réponse à l'appel et nous devons réitérer l'appel, puis ensuite une fois les avoir au téléphone de pouvoir trouver une date et une heure qui conviennent à tout le monde.

VETOTOIL a pour but de faciliter cela car après l'inscription sur le site, l'utilisateur pourra rechercher les professionnels qui sont inscrit dans son secteur, et en choisissant son domaine d'activité et ainsi de voir les premières disponibilités pour une petite urgence mais aussi la possibilité de voir par professionnel ses disponibilités aux dates souhaitées, cela permet d'être devant son téléphone ou son ordinateur et de voir directement les rdv disponibles. Le patient pourra aussi voir ses rdv en cours et ses rdv passés ainsi que le suivi de traitement s'il y en a pour son animal. Il pourra aussi ajouter des animaux à sa fiche, pour le moment chien ou chat avec une liste de race pour chacun d'entre eux prédéfini.

Pour des raisons de sécurité la société quand 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 créés ou en créer de nouveaux, tout 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 de la réalisation de VETOTOIL j'ai appliqué à moi-même la méthode AGILE, je m'appliquais des tâches journalières, des étapes et la durée de chacune des étapes pour la conception et la réalisation.

Je faisais un point le samedi pour voir l'avancée du projet ainsi que définir les tâches de 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 fonctionnel car je n'ai pas les routes pour le moment	90%

## TITRE PROFESSIONNEL DWWM

	Ok pour mobile tablette et pc	
--	-------------------------------	--

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'envoi de mail (<https://github.com/PHPMailer/PHPMailer>)

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

-Karser pour faire le RecaptchaV3 de Google pour sécuriser les envois de mail et d'inscription (<https://github.com/karser/KarserRecaptcha3Bundle>)

-UuidV7 (fonction unique) pour générer des id aléatoire et « unique » pour certaines bases de données sensibles. (<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 différentes tailles 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 informations demandées (nom, prénom, email(ne peut avoir deux emails identiques), password) ensuite à la connexion ils seront redirigés vers une autre page des compléments d'information.

#### Pour les Sociétés :

-création d'une fiche société (informations demandées : 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 créer un employé (fiche user normale)
- Doit pouvoir rechercher un employé par son email s'il est déjà inscrit
- Doit pouvoir ajouter un employé à son équipe et définir les jours travaillés 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 employeur, complément adresse employeur, code postal employeur, ville employeur, téléphone employeur, profession employeur, image employeur, date de création employeur)
- Ne peut être recherché 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 animaux (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 dégradé 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

## 2.3.1 Description 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.

## TITRE PROFESSIONNEL DWWM

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	<p>Une fois connecté, il doit pouvoir accéder aux informations suivantes :</p> <p>Rendez-vous du jour (heure, nom de la personne) pris ou libre, en cliquant sur le rdv</p> <p>Afficher les informations du patient (nom, prénom, téléphone, statistiques de % de rendez-vous validés).</p> <p>Possibilité pour le professionnel d'annuler le rendez-vous.</p> <p>Possibilité d'ajouter un rendez-vous sur un créneau libre.</p> <p>Confirmer la présence ou l'absence du patient.</p>
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

## TITRE PROFESSIONNEL DWWM

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

### 2- 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.

## TITRE PROFESSIONNEL DWWM

Niveau de priorité	Priorité Essentiel
<b>Fonction : Enregistrement Animal client</b>	
Objectif	Lister le ou les animaux du client
Description	Un client peut avoir un ou plusieurs animaux de compagnies 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

<b>Fonction : session de connexion</b>	
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

<b>Fonction : modification fiche client</b>	
Objectif	Page pour modifier supprimer la fiche client
Description	Récupérer toutes les informations client pour pouvoir la modifier ou supprimer le compte ou ajouter un animal

## TITRE PROFESSIONNEL DWWM

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

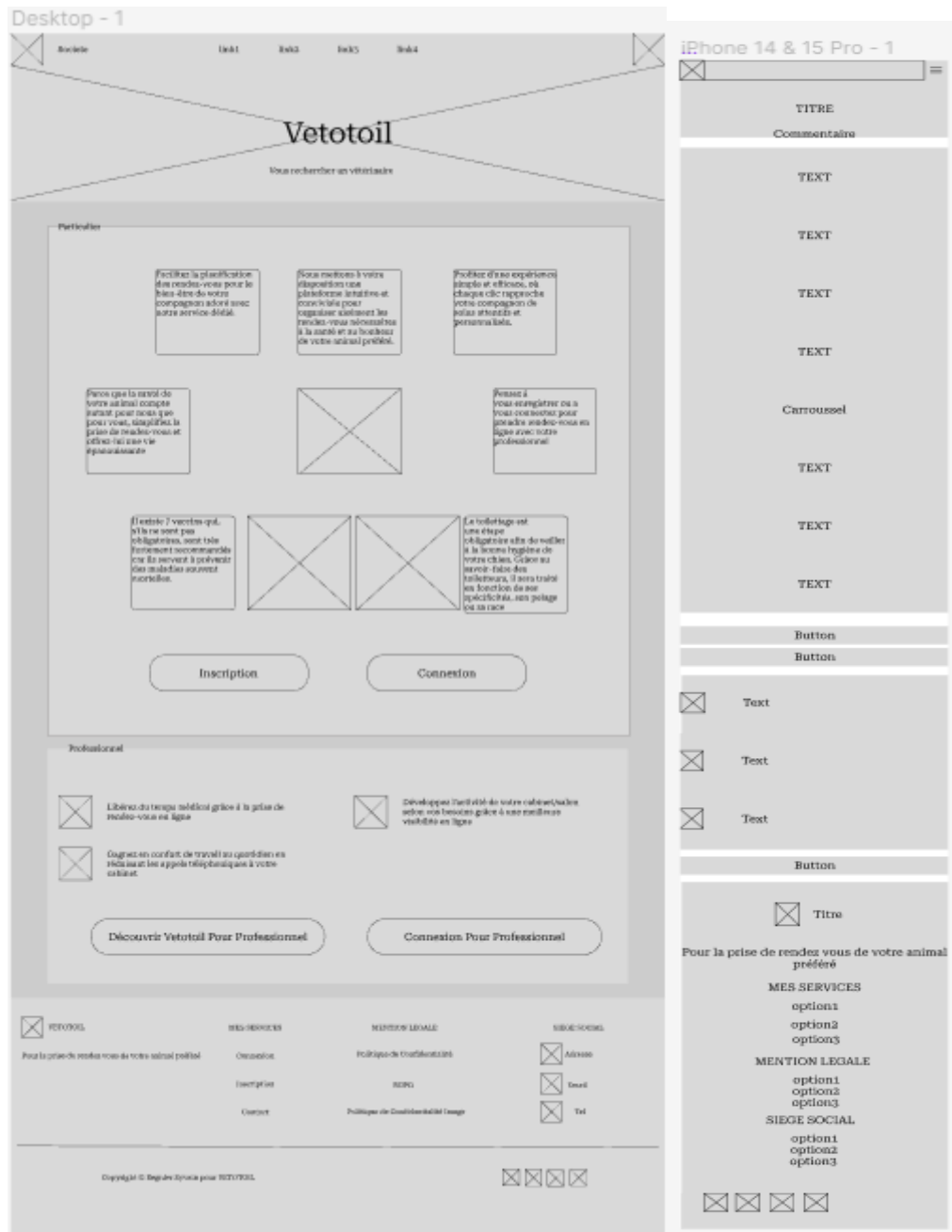
Fonction : voir ses rendez-vous en cours	
Objectif	Voir ses rendez-vous en cours
Description	Pouvoir accéder a ses rdv sur la page d'accueil quand l'utilisateur est connecté et fournir un lien aussi
Contrainte	Affiché les dates des rendez-vous chez qu'elle professionnel non adresse ect..., pouvoir annuler un rdv mais impossible de l'annuler si date inférieur a 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 ect...
Niveau de priorité	Priorité Souhaitable





## 2.4 Création des WireFrames :



## 2.5 Réalisation des maquettes :

### Version PC





### 03 Introduction et structure :

Pour la partir Front-End, j'ai décidé d'utilisé du HTML,le framework Bootstrap ,Javascript ,twig et un peu de CSS le tout fait avec le Framework Symfony et son architecture.

J'ai aussi utilisé des librairies qui sont à la fois Front-End et Back-End comme :

-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](https://symfony.com/doc/current/controller/upload_file.html))

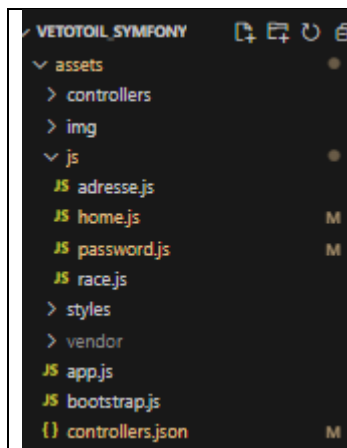
J'ai aussi utilisé l'API ADRESSE de l'Etat pour permettre de l'utilisateur de trouver celle-ci en fonction de ce qu'il va taper et ainsi renseigner la ville et le code postal automatiquement

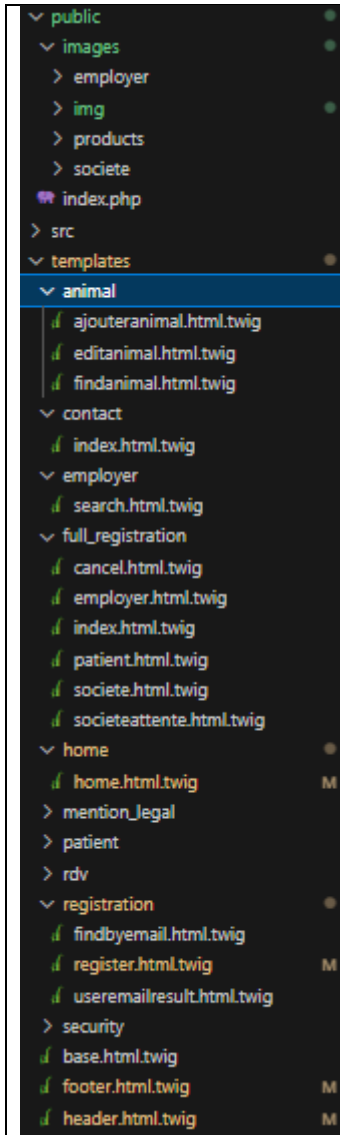
Pour pouvoir avoir un rendu en mode développement il faut au préalable lancer le serveur de Symfony avec la commande :

```
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

	<p>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</p>
------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Dans cette partie dans le dossier public, j'ai créé un dossier images dans lequel il y a 4 autres dossiers.</p> <p>Le dossier img est pour toutes les images du site</p> <p>Les 3 autres dossiers sont lorsque les utilisateurs s'inscrivent ils peuvent ajouter une photo à leurs profils via vichupload et sont trié par rapport à leurs rôles.</p> <p>Ensuite l'on peut voir le dossier templates dans lequel il y a les vues sous l'extension .html.twig qui indique que l'on peut mettre du code html dedans et le twig permet de récupérer des variables pour rendre le site plus dynamique et non statiques</p> <p>Le fichier base.html.twig est le fichier de référence pour les autres pages c'est dedans que nous allons faire appel au script Bootstrap et importmap App qui fait référence au fichier app.js dans lequel le chemin du style/app.css est déclaré, cela est fait automatiquement par Symfony</p>
------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Voici le fichier base.html.twig

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22><text y=%221.2em%22 font-size=%2296%22>
    </text><text y=%221.3em%22 x=%220.2em%22 font-size=%2276%22 fill=%22%23ff%22><svg%22>
    </text></text></svg%22>" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQ3ZazprGIAAnm3Q0gpJLIm9Nao0Yz1ztcQTwfspd3y065VohhpuuComLAsjC" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-Mrcw62MFYIzclA8N1+ntUvF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/js/bootstrap.bundle.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.1.2/font/bootstrap-icons.min.css">

    {% block stylesheets %}
    {% endblock %}

    {% block javascripts %}
    {% block importmap %}{{ importmap('app') }}{% endblock %}

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

    {% endblock %}
  </head>
  <body>

    {% include "header.html.twig" %}

    {% block body %}
    {% endblock %}

    {% include "footer.html.twig" %}
  </body>
</html>
```

Dans ce fichier nous voyons donc les liens pour accéder à la bibliothèque Bootstrap ainsi qu'à Bootstrap icon

De plus Symfony génère automatiquement le block JavaScript dans lequel il y a l'importation de ('app') ou l'on peut retrouver l'import du fichier css pour qu'il soit actif sur toutes les pages.

Ensuite toutes les pages seront traitées dans la balise <body> importer le header et le footer permet de ne plus avoir besoin de l'importer et sera visible sur toutes les pages du site.

En ce qui concerne le {% block body %} {% endblock %} cela sera à mettre dans toutes les pages pour informer le navigateur ou il doit l'affiché sur la page.

## 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.

Grace à la bibliothèque Bootstrap , le responsive est plus facile à gérer et nous avons besoin de moins coder dans le fichier css .

Voici une partie du [home.html.twig](#)

```
{% extends 'base.html.twig' %}

{% block title %}Vetotoil
{% endblock %}

{% block body %}

    <div class="row justify-content-center">
        <div class="col-12 text-center position-relative">
            
            <h1 class="position-absolute top-50 start-50 translate-middle grand" style="transform: translate(-50%, -50%); z-index: 1;">VETOTOIL</h1>
            <p class="position-absolute top-50 start-50 translate-middle mt-5" style="transform: translate(-50%, 50%); z-index: 1;">Vous recherchez un
                <span id="choix">Vétérinaire</span>
            </p>
        </div>
    </div>
    <div class="container" id="home">

{% for message in app.flashes('info') %}
    <div class="alert alert-info">
        {{ message }}
    </div>
{% endfor %}

    <fieldset class="border p-2 mb-4 mt-4 custom-box-shadow">
        <legend class="float-none w-auto">
            <strong>Particulier</strong>
        </legend>
        <div class="row justify-content-center py-5">
            <div class="col-12 col-md-3 m-md-3 mb-3 rounded-3 degrader col-lg-2">
                <p>Facilitez la planification des rendez-vous pour le bien-être de votre compagnon adoré avec notre service dédié.</p>
            </div>
            <div class="col-md-3 col-12 m-md-3 mb-3 rounded-3 degrader col-lg-2">
                <p>Nous mettons à votre disposition une plateforme intuitive et conviviale pour organiser aisément les rendez-vous nécessaires à la santé et au bonheur de votre animal préféré.</p>
            </div>
            <div class="col-md-3 col-12 m-md-3 mb-3 rounded-3 degrader col-lg-2">
                <p>Profitez d'une expérience simple et efficace, où chaque clic rapproche votre compagnon de soins attentifs et personnalisés.</p>
            </div>
        </div>
        <div class="row justify-content-between">
            <div class="col-md-3 col-12 m-md-3 mb-3 rounded-3 degrader col-lg-2">
                <p>Parce que la santé de votre animal compte autant pour nous que pour vous, simplifiez la prise de rendez-vous et offrez-lui une vie épanouissante</p>
            </div>
        </div>
    </div>

```

Dans un premier temps on étend le fichier « base.html.twig » pour permettre d'avoir l'affichage prédéfini (pour mon cas cela évite de remettre les liens Bootstrap ,header... ) , puis ensuite nous mettons le code dans la balise {% block body %} et à la fin de la page il faut refermer cette balise avec {% endblock %}

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 peut excéder 12 colonnes et chaque colonne peut encore avoir 12 colonnes.

Cette ligne veut dire qu'il y a une ligne et qu'elle est centrée 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 centré 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">

```

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;
    }

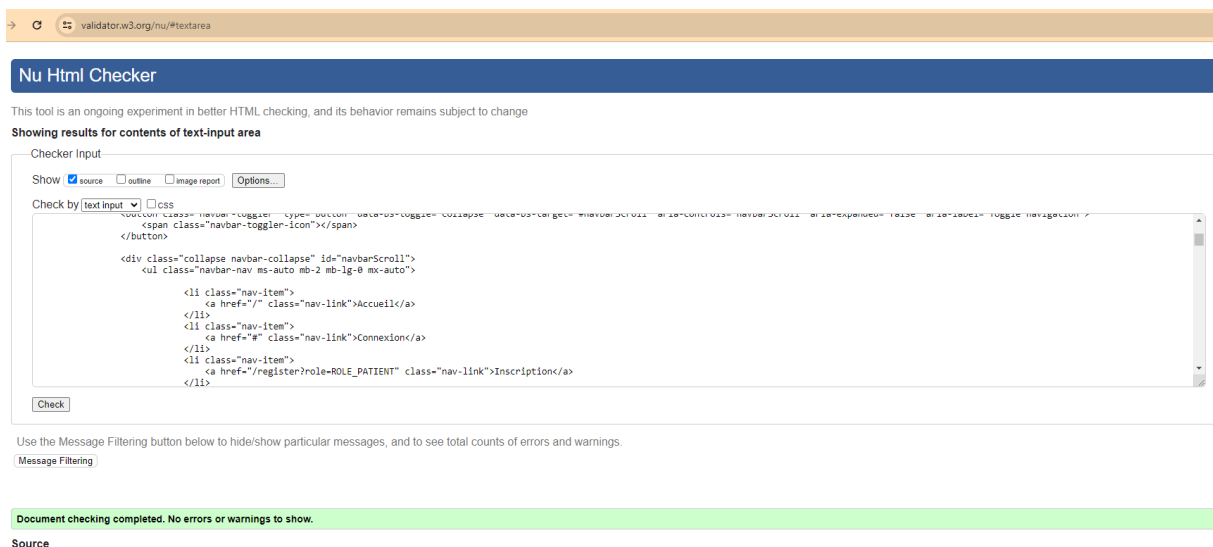
    .grand{
        font-size: 500%;
    }

}
```

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 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

The image displays two side-by-side screenshots of a web registration form titled "Inscription".

**Left Screenshot:** Shows the initial state of the form with input fields for Nom, Prenom, Email, Password, Repeat Password, and a checkbox for "Agree terms". A "Valider" button is at the bottom. A purple header bar is visible at the top.

**Right Screenshot:** Shows the form after the "Password" field has been interacted with. A tooltip message appears above the Password field stating: "localhost:8000 indique Il faut au minimum 8 caractère". Below the "Repeat Password" field, there are two green checkmark icons and a list of password requirements:

- Mettre des lettres et de chiffres
- Une ou plusieurs Majuscule
- 8 Caractères minimum
- 1 Caractere special

The "Agree terms" checkbox and the "Valider" button are also visible in this state.

```

console.log("password chargé");
document.addEventListener("DOMContentLoaded", function () {
  let verification = document.getElementById("registration_form_submit");
  verification.addEventListener("click", function(event){
    let mdp = document.getElementById("registration_form_plainPassword_first").value;
    let remdp = document.getElementById("registration_form_plainPassword_second").value;
    let errors = [];

    if (mdp !== remdp) {
      errors.push('Les mots de passe saisis sont différents.');
```

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 déclarée errors en tableau et qui est push (insérer) au fur et à mesure des erreur détecté

```

let mdp = document.getElementById("registration_form_plainPassword_first");
let chiffre = document.getElementById("chiffre");
let majuscule = document.getElementById("majuscule");
let caractere = document.getElementById("caractere");

mdp.addEventListener('input', function() {
  let specialDocument = document.getElementById("special");
  let mdpValue = mdp.value;
  if (mdpValue.length > 0) {
    ulVisible.classList.remove("d-none");

    let expression = /\d/;
    let regex1 = /[a-z]/;
    let special = /[!@#%&*()_+~\-=\[\]{};':"\"|,.<>\/?]/;
    if (expression.test(mdpValue) && regex1.test(mdpValue)) {
      chiffre.classList.remove("d-none");
    } else {
      chiffre.classList.add("d-none");
    }

    if (mdpValue.match(/[A-Z]/, 'g')) {
      majuscule.classList.remove("d-none");
    } else {
      majuscule.classList.add("d-none");
    }

    if (mdpValue.length > 7) {
      caractere.classList.remove("d-none");
    } else {
      caractere.classList.add("d-none");
    }

    if (special.test(mdpValue)) {
      specialDocument.classList.remove("d-none");
    } else {
      specialDocument.classList.add("d-none");
    }
  } else {
    ulVisible.classList.add("d-none");
  }
});
```

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

## TITRE PROFESSIONNEL DWWM

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

Information sur la société	
<div><div>Siret</div><div></div><div>Rechercher</div></div> <div><div>Nom de la Société</div><div></div></div> <div><div>Profession</div><div>Choisissez...</div></div> <div><div>Adresse</div><div></div></div> <div><div>Complement Adresse</div><div></div></div> <div><div>Code Postal</div><div></div></div> <div><div>Ville</div><div></div></div> <div><div>Téléphone société</div><div></div></div>	<div><div>Siret</div><div>39860733300059</div><div>Rechercher</div></div> <div><div>Nom de la Société</div><div>CHAPO LA PAILLE</div></div> <div><div>Profession</div><div>Choisissez...</div></div> <div><div>Adresse</div><div>44 RUE NANTIER DIDIE</div></div> <div><div>Complement Adresse</div><div></div></div> <div><div>Code Postal</div><div>97490</div></div> <div><div>Ville</div><div>SAINT-DENIS</div></div> <div><div>Téléphone société</div><div></div></div>

```

console.log("siret chargé");

document.addEventListener("DOMContentLoaded", function () {
  const url = "https://api.insee.fr/entreprises/sirene/V3/siret/";
  const accessToken = "75c7b088-1c1c-3cdd-8ce2-dba8a65362df";
  const button = document.getElementById("validerSiret");
  button.addEventListener("click", async function () {
    const siret = document.getElementById("societe_siret");
    // console.log("test : ", siret.value);

    siret
      .addEventListener("keydown", function (event) {
        let keyCode = event.which || event.keyCode;

        //autoriser que les chiffres et certaines touche comme supprimer
        //=====
        if (
          (keyCode >= 48 && keyCode <= 57) ||
          (keyCode >= 96 && keyCode <= 105) ||
          [8, 9, 13, 27, 46].includes(keyCode)
        ) {
          event.preventDefault();
        }
        //=====
        //limite a 14chiffres et supprime les lettres
        //=====
        let inputValue = event.target.value.replace(/[^0-9]/g, "");
        if (inputValue.length >= 14 && ![8, 46].includes(keyCode)) {
          event.preventDefault();
        }
      });

      //=====
      //recherche ds l api siret de l insee les info
      //=====
      const headers = {
        "Content-Type": "application/x-www-form-urlencoded",
        Accept: "application/json",
        Authorization: "Bearer ${accessToken}",
      };
      try {
        const response = await fetch(url + siret.value, {
          method: "GET",
          headers: headers,
        });
        if (!response.ok) {
          throw new Error("Réponse réseau non OK");
        }
        const resultat = await response.json();
        // console.log("resultat : ", resultat);
        searchInfo(resultat);
      } catch (error) {
        console.error("Erreur :", error);
      }
    });

    function searchInfo(resultat) {
      const numeroVoie =
        resultat.etalblissement.adresseEtablissement.numeroVoieEtablissement;
      const typeVoie =
        resultat.etalblissement.adresseEtablissement.typeVoieEtablissement;
      const libelleVoie =
        resultat.etalblissement.adresseEtablissement.libelleVoieEtablissement;
      const complementAdresse =
        resultat.etalblissement.adresseEtablissement
          .complementAdresseEtablissement;
      const libelleCommune =
        resultat.etalblissement.adresseEtablissement.libelleCommuneEtablissement;
      const codePostal =
        resultat.etalblissement.adresseEtablissement.codePostalEtablissement;
      const denominationUsuelleEtablissement =
        resultat.etalblissement.periodesEtablissement[0]
          .denominationUsuelleEtablissement;
      // console.log('denominationUsuelleEtablissement : ', denominationUsuelleEtablissement);
      const nomUniteLegale = resultat.etalblissement.uniteLegale.nomUniteLegale;
      document.getElementById("societe_adresse_societe").value =
        numeroVoie + " " + typeVoie + " " + libelleVoie;
      document.getElementById("societe_complement_adresse_societe").value = complementAdresse;
      document.getElementById("societe_code_postal_societe").value = codePostal;
      document.getElementById("societe_ville_societe").value = libelleCommune;
    }
  });
});

```

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

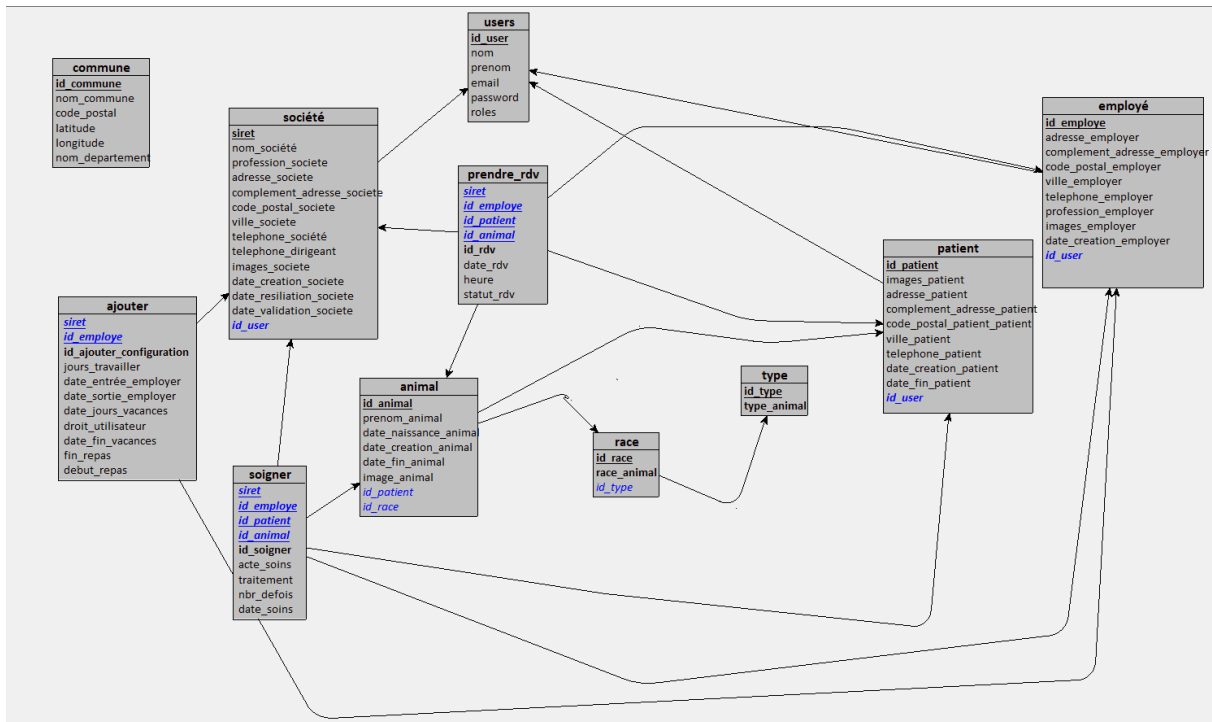
Je récupère en json le résultat

Je récupère les informations  
nécessaires

J'insère les information  
récupéré dans l'input



## 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 fonction 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

	<p>Dans le dossier src</p> <p>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</p> <p>DataFixture pour intégrer les données fixe ou fake dans les tables.</p> <p>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</p> <p>Le Form permet de crée, traiter et valider des formulaires plus rapidement, ils peuvent être réutilisable sur différente vues</p> <p>Le dossier fonction que j'ai créé pour mettre certaine fonction de traitement réutilisable</p> <p>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</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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 créer la base de données nous avons besoin de Mysql utilisé avec phpMyAdmin pour la visualisation, j'ai utilisé celui de XAMPP.

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

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

Et j'ai rajouté la ligne pour dire à Symfony comment se connecter à la base de données

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/vetotoil_symfony?charset=utf8"
```

Mysql est gestionnaire de base de données utilisé, root et le nom d'utilisateur pour me connecter à la base de données, root est par défaut mais il peut être différent si lors de la configuration à la base de données nous avons modifié 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 données et le jeu de caractères qui permet une compatibilité entre les différentes langues et caractères.

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

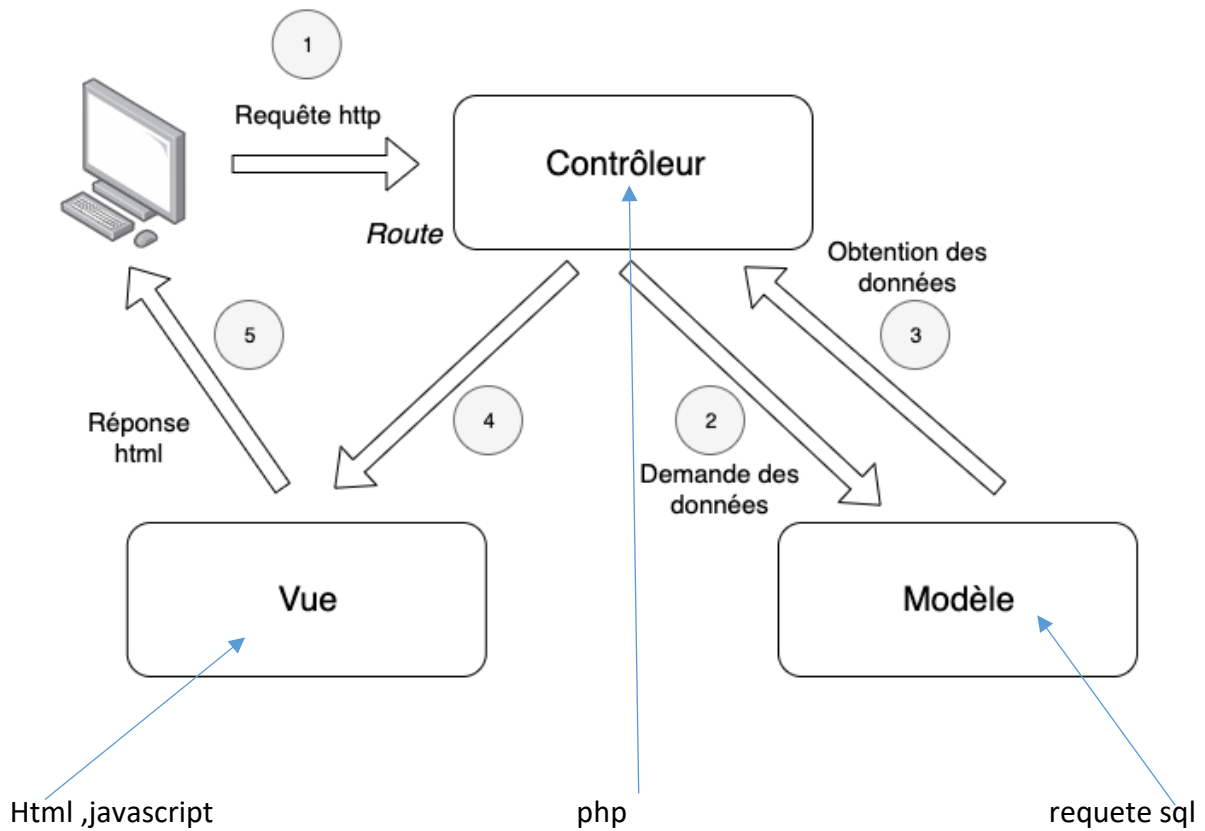
```
php bin/console doctrine:database:create
```

Ceci crée la base de données mais sans table à l'intérieur

#### 4.2.3 Sous le MVC

Dans un premier temps voici le fonctionnement du MVC la requête http fait appel à une route qui redirige vers le contrôleur celui-ci suivant l'action à effectuer peut soit retourner une vue qui sera affichée à l'écran de l'utilisateur soit faire une demande de données au

modèle qui en retour reçoit une réponse qui ensuite le retourne à la vue ;



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

```

38 references
private $bd;

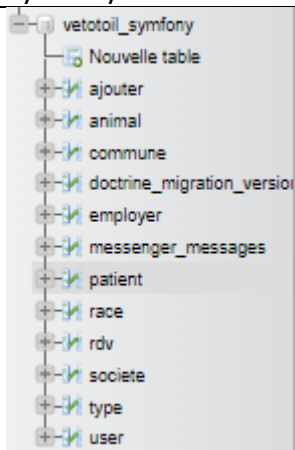
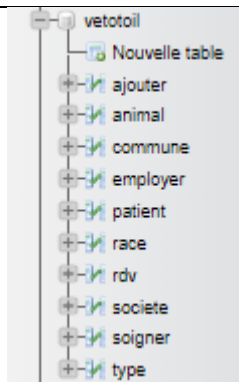
3 references
private static $instance = null;

1 reference
private function __construct()
{
    try {
        $this->bd = new PDO('mysql:host=localhost;dbname=vetotoil', 'root', '');
        $this->bd->query("SET NAMES 'utf8'");
        $this->bd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        die('<p>Echec connexion. Erreur [' . $e->getCode() . ']: ' . $e->getMessage() . '</p>');
    }
}

```

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
```

## TITRE PROFESSIONNEL D'WWM

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

```
PS C:\Users\sregnier\Documents\projets-afpa\symfony_vetotoils> php bin/console make:entity

Class name of the entity to create or update (e.g. BrandPuppy):
> Vetotoil

Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:
>

created: src/Entity/Vetotoil.php
created: src/Repository/VetotoilRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command:

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

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

Field length [255]:
> 50

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

updated: src/Entity/Vetotoil.php

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

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

```

class Version2020011102036 extends AbstractMigration
{
    @before
    public function getDescription(): string
    {
        return '';
    }

    @after
    public function migrate(Schema $schema): void
    {
        // this migration is auto-generated, please modify it to your needs

        $this->addSql('CREATE TABLE ajutor (id BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', societe_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=societe)\', employee_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=employee)\', date_entree_employee DATE NOT NULL, date_sortie_employee DATE NOT NULL, date_animal BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', race_id INT DEFAULT NULL, user_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=user)\', prenom_animal VARCHAR(50) NOT NULL, date_naissance_animal DATE DEFAULT NULL, date_creation_animal DATE NOT NULL, CREATE TABLE animal (id INT AUTO_INCREMENT NOT NULL, nom_comune VARCHAR(180) NOT NULL, code_postal VARCHAR(10) NOT NULL, longitude VARCHAR(50) NOT NULL, latitude VARCHAR(50) NOT NULL, nom_departement VARCHAR(180) NOT NULL, PRIMARY KEY(id) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ENGINE = InnoDB);
        $this->addSql('CREATE TABLE employeur (id BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', user_id BINARY(16) NOT NULL COMMENT \'(DC2Type=user)\', adresse_employeur VARCHAR(255) NOT NULL, complement_adresse_employeur VARCHAR(255) DEFAULT NULL, code_postal_employeur VARCHAR(10) NOT NULL, CREATE TABLE patient (id BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', adresse_patient VARCHAR(255) NOT NULL, complement_adresse_patient VARCHAR(255) DEFAULT NULL, code_postal_patient VARCHAR(10) NOT NULL, CREATE TABLE race (id INT AUTO_INCREMENT NOT NULL, type_id INT DEFAULT NULL, type_animal VARCHAR(80) NOT NULL, INDEX IDX_5D68F854C83AC8C9 (type_id), PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci ENGINE = InnoDB);
        $this->addSql('CREATE TABLE rsv (id INT AUTO_INCREMENT NOT NULL, societe_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=societe)\', animal_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=animal)\', employee_id BINARY(16) DEFAULT NULL COMMENT \'(DC2Type=employee)\', patient_id BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', user_id BINARY(16) NOT NULL COMMENT \'(DC2Type=user)\', adresse_vache VARCHAR(180) NOT NULL, nom_societe VARCHAR(180) NOT NULL, profession_societe VARCHAR(28) NOT NULL, adresse_societe VARCHAR(255) NOT NULL, CREATE TABLE user (id BINARY(16) NOT NULL COMMENT \'(DC2Type=id)\', email VARCHAR(180) NOT NULL, roles JSON NOT NULL COMMENT \'(DC2Type=json)\', password VARCHAR(255) NOT NULL, nom VARCHAR(50) NOT NULL, prenom VARCHAR(50) NOT NULL, UNIQUE INDEX UNIQ_80393046E797F3E0 (email) ON DUPLICATE KEY UPDATE nom=nom, prenom=prenom, UNIQUE INDEX UNIQ_80393046E797F3E0 (password) ON DUPLICATE KEY UPDATE email=email, roles=roles, created_at=DATETIME NULL, created_at DATETIME NOT NULL COMMENT \'(DC2Type=datetime)\', available_at DATETIME NOT NULL COMMENT \'(DC2Type=datetime)\',
        $this->addSql('ALTER TABLE ajutor ADD CONSTRAINT FK_A3B8859F7C77F780 FOREIGN KEY (employee_id) REFERENCES employeur (id)');
        $this->addSql('ALTER TABLE ajutor ADD CONSTRAINT FK_A3B8859F41C0267A FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_A3A231F87AED9195 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_A3A231F87AED9195 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE employeur ADD CONSTRAINT FK_D2C4CF786678D935 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE patient ADD CONSTRAINT FK_1A20787A7B91D935 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE patient ADD CONSTRAINT FK_D0AF88F5C4C8C93 FOREIGN KEY (type_id) REFERENCES type (id)');
        $this->addSql('ALTER TABLE rsv ADD CONSTRAINT FK_D2C4CF786678D935 FOREIGN KEY (societe_id) REFERENCES societe (id)');
        $this->addSql('ALTER TABLE rsv ADD CONSTRAINT FK_1C1C31F86E2C16 FOREIGN KEY (animal_id) REFERENCES animal (id)');
        $this->addSql('ALTER TABLE rsv ADD CONSTRAINT FK_1C1C31F86E2C16 FOREIGN KEY (employee_id) REFERENCES employeur (id)');
        $this->addSql('ALTER TABLE rsv ADD CONSTRAINT FK_1C1C31F86E2C16 FOREIGN KEY (patient_id) REFERENCES patient (id)');
        $this->addSql('ALTER TABLE societe ADD CONSTRAINT FK_13610380A8D92D35 FOREIGN KEY (user_id) REFERENCES user (id)');

        // this down() migration is auto-generated, please modify it to your needs

        public function down(Schema $schema): void
        {
            // this down() migration is auto-generated, please modify it to your needs

            $this->addSql('ALTER TABLE ajutor DROP FOREIGN KEY FK_A3B8859F7C77F780');
            $this->addSql('ALTER TABLE ajutor DROP FOREIGN KEY FK_A3B8859F41C0267A');
            $this->addSql('ALTER TABLE animal DROP FOREIGN KEY FK_A3A231F87AED9195');
            $this->addSql('ALTER TABLE animal DROP FOREIGN KEY FK_A3A231F87AED9195');
            $this->addSql('ALTER TABLE employeur DROP FOREIGN KEY FK_D2C4CF786678D935');
            $this->addSql('ALTER TABLE patient DROP FOREIGN KEY FK_1A20787A7B91D935');
            $this->addSql('ALTER TABLE patient DROP FOREIGN KEY FK_D0AF88F5C4C8C93');
            $this->addSql('ALTER TABLE rsv DROP FOREIGN KEY FK_D2C4CF786678D935');
            $this->addSql('ALTER TABLE rsv DROP FOREIGN KEY FK_1C1C31F86E2C16');
            $this->addSql('ALTER TABLE rsv DROP FOREIGN KEY FK_1C1C31F86E2C16');
            $this->addSql('ALTER TABLE societe DROP FOREIGN KEY FK_13610380A8D92D35');
            $this->addSql('DROP TABLE ajutor');
        }
    }
}

```

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
```

```
<?php
namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Uid\UuidV7;

#[ORM\Entity(repositoryClass: UserRepository::class)]
#[UniqueEntity(fields: ['email'], message: 'There is already an account with this email')]
36 references | 0 implementations
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue('CUSTOM')]
    #[ORM\Column(type: 'uuid', unique: true)]
    #[ORM\CustomIdGenerator('doctrine.uuid_generator')]
    1 reference
    private ?UuidV7 $id = null;

    #[ORM\Column(length: 180, unique: true)]
    3 references
    private ?string $email = null;

    /**
     * @var list<string> The user roles
     */
    #[ORM\Column]
    2 references
    private array $roles = [];

    /**
     * @var string The hashed password
     */
    #[ORM\Column]
    2 references
    private ?string $password = null;

    #[ORM\Column(length: 50)]
    2 references
    private ?string $nom = null;
}
```

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

<pre> #[ORM\OneToMany(targetEntity: Animal::class, mappedBy: 'user')] 5 references private Collection \$animals; #[ORM\OneToOne(mappedBy: "user", cascade: ['persist', 'remove'])]  2 references private ?Societe \$societe = null;  #[ORM\OneToOne(mappedBy: "user", cascade: ['persist', 'remove'])] 2 references private ?Employer \$employer = null;  1 reference   0 overrides public function __construct() {     \$this-&gt;animals = new ArrayCollection(); }  0 references   0 overrides public function getId(): ?UuidV7 {     return \$this-&gt;id; }  0 references   0 overrides public function getEmail(): ?string {     return \$this-&gt;email; }  0 references   0 overrides public function setEmail(string \$email): static {     \$this-&gt;email = \$email;     return \$this; }  /**  * A visual identifier that represents this user.  *  * @see UserInterface  */ 0 references   0 overrides public function getUserIdentifier(): string {     return (string) \$this-&gt;email; } </pre>	<p>Il est aussi précisé le type de relation entre les entity dans ce cas l'entity user car un user peut avoir plusieurs animaux car c'est du OneToMany, mais un animal ne peut avoir plusieurs propriétaires</p> <p>Sur cette partie nous avons les getters (lire) Ici nous avons les setters (envoyer)</p> <p>Il crée aussi un champs rôles (droit pour les utilisateurs du site) automatiquement Ensuite il faut définir les rôles dans config/packages/security.yaml</p> <pre> access_control:     - { path: ^/patient, roles: ROLE_PATIENT }     - { path: ^/societe, roles: ROLE_SOCIETE }     - { path: ^/employer, roles: ROLE_EMPLOYER }     - { path: ^/administrateur, roles: ROLE_ADMINISTRATEUR } </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Maintenant pour pouvoir utiliser cette entity j'ai besoin de créer un contrôleur avec la commande

```
php bin/console make:controller
```

<pre> PS C:\Users\sregnier\Documents\projets-afpa\symfony_vetotoil5&gt; php bin/console make:controller  Choose a name for your controller class (e.g. GrumpyPizzaController): &gt; AnimalController  created: src/Controller/AnimalController.php created: templates/animal/index.html.twig  Success!  Next: Open your new controller class and add some pages! PS C:\Users\sregnier\Documents\projets-afpa\symfony_vetotoil5&gt; </pre>	<p>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.</p> <p>Renseigne le nom</p> <p>Puis symfony crée le contrôleur mais aussi un</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

dossier du nom du Controller avec la vue index.html.twig

Voici le controller crée

```
<?php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class AnimalController extends AbstractController
{
    #[Route('/animal', name: 'app_animal')]
    public function index(): Response
    {
        return $this->render('animal/index.html.twig', [
            'controller_name' => 'AnimalController',
        ]);
    }
}
```

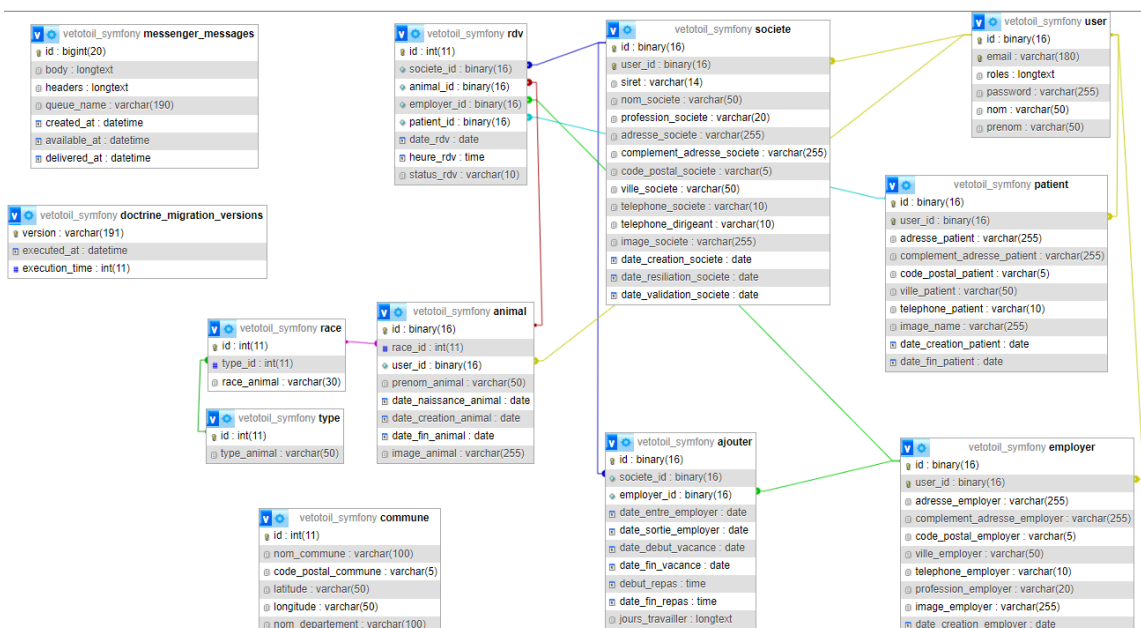
Ici on peut voir que symfony extends la class animal à AbstractController en réalité on hérite de la class crée par symfony pour simplifier le code et réduit le code répétitif

Il crée aussi une route par défaut pour notre controller 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

## 4.3.1.1 Modèle 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 contrôleurs ainsi que des modèles et des autres fichiers.

```
<?php

require_once('Models/Model.php');
require_once('Controllers/Controller.php');
require_once('Utils/header.php');
require_once('Utils/function_validation_text.php');

$controllers=['home','acces','session','pro','animal','rdv','employer'];
$controller_default='home';

if(isset($_GET['controller']) and in_array($_GET['controller'],$controllers))
{
    $nom_controller=$_GET['controller'];
}
else
    $nom_controller=$controller_default;

$nom_classe="Controller_".$nom_controller;
$nom_fichier="Controllers/".$nom_classe.".php";

if(file_exists($nom_fichier))
{
    require_once($nom_fichier);
    $controller=new $nom_classe();
}
else
    exit("ERROR 404: not found");

require_once('Utils/footer.php');

?>
```

Les `require_once` permettent de charger qu'une seule fois la page demandée sans avoir à la recharger constamment ici ce sont le header le contrôleur qui va gérer le fonctionnement des autres Controller et le modèle

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 `Controller_` car tous les noms des contrôleurs commencent par ceci puis en ajoutant la variable récupérée préalablement pour ensuite accéder au contrôleur



```

<?php
7 references | 7 implementations
abstract class Controller //Ceci est le controleur par défaut
{
    1 reference | 7 overrides
    abstract public function action_default();

    0 references | 0 overrides
    public function __construct()
    {
        if (isset($_GET['action']) and method_exists($this, "action_" . $_GET["action"])) {
            $action = "action_" . $_GET['action'];
            $this->$action(); //On appelle cette action
        } else $this->action_default(); //sinon action par défaut
    }

    51 references | 0 overrides
    protected function render($vue, $data = []) //Fonction qui recupere les données et les transmet a la vu
    {
        extract($data); //Recupération des données à afficher

        if (isset($_GET['controller'])) {
            $controller_actif = ucfirst($_GET['controller']);
        } else {
            $controller_actif = "Home";
        }

        $file_name = "Views/" . $controller_actif . "/view_" . $vue . ".php";
        if (file_exists($file_name)) { //Si le fichier existe
            require($file_name); //Si oui on l'affiche
        } else {
            $this->action_error("La vue n'existe pas !");
        }
    }

    1 reference | 0 overrides
    protected function action_error($message) // en cas d'erreur
    {
        $data = ['erreur' => $message];
        $this->render('error', $data);
        die(); //Pour faire terminer le script vu qu'il y a une erreur
    }
}

```

Ceci est le controleur 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

```
<?php
0 references | 0 implementations
class Controller_employeur extends Controller
{
    1 reference | 0 overrides | prototype
    public function action_default()
    {
        $this->action_home();
    }

    1 reference | 0 overrides
    public function action_home()
    {
        $this->render('home');
    }

    0 references | 0 overrides
    public function action_connexion(){
        $this->render('connexion');
    }

    0 references | 0 overrides
    public function action_session_connect_employeur(){
        $m=Model::get_model();
        $connectEmployer=['connectEmployer'=>$m->get_connexion_employeur($_POST)];

        $this->render('session',$connectEmployer);
    }

    0 references | 0 overrides
    public function action_connexion_ok_employeur()
    {
        $m = Model::get_model();
        $idEmployer=$_SESSION['id'];
        $rdvs=['rdvs'=>$m->get_mes_rdv($idEmployer)];
        $this->render('home_employeur',$rdvs);
    }
}
```

On fait appel à la class qui est étendu à Controller

Sur la fonction action\_home lorsque l'on appelle ce Controller avec pour action home on retourne sur la vue home.

Sur la fonction action\_session\_connect\_employeur 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 fonction get\_connexion\_employeur 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

```

1 reference | 0 overrides
public function get_connexion_employer(array $data)
{
    try {
        $email = $_POST['email'];
        $password = $_POST['password'];

        $requete = $this->bd->prepare('SELECT * FROM employer WHERE email=:email');
        $requete->execute(array(':email' => $email));
        $requete = $requete->fetch(PDO::FETCH_ASSOC);

        if ($requete && password_verify($password, $requete['password'])) {
            return $requete;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        die("Erreur [" . $e->getCode() . "]: " . $e->getMessage());
    }
}

```

Dans cette fonction on commence à récupérer 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 retourner 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 renvoie au controller le résultat de \$requete tandis que si c'est faux il renvoie 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

`SELECT * FROM "rdv"`

☐ Profilage [\[ Éditer en ligne \]](#) [\[ Éditer \]](#) [\[ Expliquer SQL \]](#) [\[ Créer le code source PHP \]](#) [\[ Actualiser \]](#)

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes:  | Trier par clé : Aucun(e)

Options supplémentaires

		id_rdv	date_rdv	heure	status_rdv	id_employeur	id_patient	siret	id_animal
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	1	2024-02-16	11:00:00	1	11	211	12341564897897	NULL
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	2	2024-02-26	10:00:00	NULL	13	242	12345678912345	NULL
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	12	2024-02-26	09:00:00	Annuler	13	242	12345678912345	NULL
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	29	2024-02-26	12:00:00	Annuler	13	242	12345678912345	15
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	30	2024-02-27	09:00:00	Annuler	15	242	23232323232323	14
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	31	2024-02-23	09:00:00	Annuler	21	242	97979797979797	14
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	32	2024-02-23	10:00:00	Valider	21	211	97979797979797	14
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	33	2024-02-26	09:00:00	NULL	21	242	97979797979797	14
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	34	2024-02-23	11:00:00	NULL	13	242	97979797979797	14
<input type="checkbox"/>	<a href="#">Éditer</a> <a href="#">Copier</a> <a href="#">Supprimer</a>	36	2024-03-12	11:00:00	NULL	22	242	97979797979797	13

```

public function get_voir_rdv_mois($idEmployer){
    try{
        $requete=$this->bd->prepare("SELECT *
        FROM rdv
        INNER JOIN animal ON rdv.id_animal=animal.id_animal
        INNER JOIN ajouter ON rdv.id_employeur=ajouter.id_employeur
        INNER JOIN race ON animal.id_animal=race.id_race
        INNER JOIN patient ON rdv.id_patient=patient.id_patient
        WHERE rdv.id_employeur = :idEmployeur
        AND date_rdv BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 31 DAY) AND ajouter.date_sortie_employeur IS NULL" );
        $requete->execute([':idEmployeur'=>$idEmployer]);
        return $requete->fetchAll(PDO::FETCH_ASSOC);
    }catch (PDOException $e) {
        die('Erreur [ ' . $e->getCode() . ' ] : ' . $e->getMessage());
    }
}

```

Dans la table rdv j'ai les clef étrangère suivante id\_employeur , id \_patient, le Siret de l'entreprise car un employé peut travailler dans plusieurs entreprises(mais cette requête est pour le gestionnaire de la société) et j'ai aussi besoin de l'id animal car un patient peut 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 récupérer 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 récupérer les information de la table patient le tout sur la condition de recherche que rdv.id\_employeur corresponde a l idEmployeur récupéré précédemment et dans cette requete

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

```
class FullRegistrationController extends AbstractController
{
    4 references
    private $security;

    2 references | 0 overrides
    public function __construct(Security $security)
    {
        $this->security = $security;
    }
    #[Route('/full/registration', name: 'app_full_registration')]
    8 references | 0 overrides
    public function index(Security $security, PatientRepository $patientRepository, SocieteRepository $societeRepository,
        EmployerRepository $employerRepository, EntityManagerInterface $entityManager, Request $request): Response
    {
        $user = $security->getUser();

        if ($user && in_array('ROLE_PATIENT', $user->getRoles())) {
            $patient = $patientRepository->findOneBy(['user' => $user]);
            if ($patient) {
                if ($patient->getDateFinPatient() === null) {
                    return $this->redirectToRoute('app_home');
                } else {
                    $this->addFlash('info', 'Votre compte est désactivé');
                    return $this->redirectToRoute('app_logout');
                }
            } else {
                $user = new Patient();
                $user->setDateCreationPatient(new \DateTime());
                $users = $this->security->getUser();
                $user->setUser($users);
                $form = $this->createForm(PatientType::class, $user);
                $form->handleRequest($request);
                if ($form->isSubmitted() && $form->isValid()) {
                    $entityManager->persist($user);
                    $entityManager->flush();
                    return $this->redirectToRoute('app_home', ['patient' => $patient,]);
                }
            }
            return $this->render('full_registration/patient.html.twig', [
                'controller_name' => 'FullRegistrationController',
                'form' => $form->createView(),
            ]);
        }
    }
}
```

Au départ je crée un constructeur qui fait appel a Security bundle de symfony ,cela me permet de récupérer les information de l'utilisateur 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 peut y avoir qu'un seul utilisateur comme vu précédemment par rapport à l'adresse email d'inscription.

Ensuite 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 connecter.

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

```
$user = new Patient();
```

J'envoie la date du jour à \$user

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

Ses deux lignes me permettent de passer l'id user

Et de l'envoyer à \$user

```
$users = $this->security->getUser();  
$user->setUser($users);
```

Ensuite je fais appel au formulaire créé par rapport à 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 données à remplir

ensuite on vérifie que si le formulaire est envoyé et qu'il est valide on appelle l'EntityManagerInterface qui permet d'interagir avec les entity pour gérer 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 envoyée par le formulaire avec \$entityManager->persist(\$user)

Pour le flush c'est l'exécution du persist c'est à ce moment que les données sont redirigées

Et pour finir je retourne l'utilisateur sur la page d'accueil en envoyant le paramètre « 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
```

```
class ContactController extends AbstractController
#[Route('/contact', name: 'app_contact')]
4 references|0 overrides
public function index(Request $request): Response
{
    $form = $this->createForm(ContactType::class);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $formData = $form->getData();

        $mail = new PHPMailer(true);
        $secretKeyGoogle = $_ENV['SECRET_KEY_GOOGLE'];
        $secretEmailGoogle = $_ENV['SECRET_EMAIL_GOOGLE'];
        $secretEmailSend = $_ENV['SECRET_EMAIL_SEND'];

        try {
            $mail->isSMTP();
            $mail->Host = 'smtp.gmail.com';
            $mail->Port = 587;
            $mail->SMTPSecure = 'tls';
            $mail->SMTPAuth = true;
            $mail->Username = $secretEmailGoogle;
            $mail->Password = $secretKeyGoogle;

            $mail->setFrom($secretEmailGoogle);
            $mail->addAddress($secretEmailSend);

            $mail->isHTML(true);
            $mail->Subject = $formData['sujet'];
            $mail->Body = 'Email: ' . $formData['email'] . '<br>Nom: ' . $formData['nom'] . '<br>Prénom: ' . $formData['prenom'] . '<br>Téléphone: ' . $formData['telephone'] . '<br>Message: ' . $formData['message'];
            $mail->AltBody = strip_tags($mail->Body);

            $mail->send();

            $this->addFlash('info', 'Votre message a été envoyé.');
```

```
return $this->redirectToRoute('app_home');
        } catch (Exception $e) {
            $this->addFlash('error', "Le message n'a pas été envoyé : {$mail->ErrorInfo}");
        }
    }

    return $this->render('contact/index.html.twig', [
        'controller_name' => 'ContactController',
        'form' => $form->createView(),
    ]);
}
```

Dans un premier temps je crée le formulaire et j'appelle 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 à l'objet PHPMailer puis je récupère les mes clefs secrètes insérées dans le `.env` de symfony.

Une fois ce traitement effectué étant donnée que j'utilise google j'ai du créer sur mon adresse email de google une clé sécurisée par rapport à mon mot de passe. Étant donnée que c'est pour l'envoi je configure le smtp de google le port utilisé le mode de sécurisation pour le chiffrement du mail (tls ou ssl) et active l'authentification du smtp, ensuite PHPMailer demande le nom d'utilisateur qui est l'adresse email et la clé secrète générée par google.

Puis on définit 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 corps du message est de l'html .

Puis on renseigne les autres champs qui seront vus dans le mail le sujet que nous récupérons grâce à la variable `$formData['sujet']` et la même procédure est utilisée 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 la page d'accueil avec la fonction `addFlash` et si il y a une erreur il y aura un message d'erreur

## 4.5 La sécurité :

Depuis toujours la sécurité est très importante dans tous types de programmes.

### 4.5.1 Les mots de passe

Dans un premier temps j'ai sécurisé les mots de passe avec la fonction de symfony intégrée

	id	email	roles	password	nom	prenom
<input type="checkbox"/>	0x018e3bd1622c7386a49e6d8589fd326	n@n.fr	["ROLE_SOCIETE"]	\$2y\$13\$NldX7DjC3NE9oHq6e7gXt.dadzX83oSiKtSrWpSf9JW...	regnier	sylvain
<input type="checkbox"/>	0x018e3bd34f77cfb8b59fa3b8474151b	toto@toto.fr	["ROLE_EMPLOYER"]	\$2y\$13\$KtUfNaRD6u9epDdg6bJqL.qB2WYVzmR7NIGYtjFqtZS...	toto	toto
<input type="checkbox"/>	0x018e3bd5d01a7aff8abf40bdd022d79d	a@a.fr	["ROLE_PATIENT"]	\$2y\$13\$wC8cyqZBS6qeJysnh9/WCeQyK9x6KkfA5gcYgO3Tn7p...	a	a
<input type="checkbox"/>	0x018e3c07f26e716389396556b14e5e49	w@w.fr	["ROLE_EMPLOYER"]	\$2y\$13\$FhamGDI8tsY2vImC9ASseO4C1Fs0gqCLDFleLUqFca...	w	wa
<input type="checkbox"/>	0x018e506a101f7d378952d4384f78142f	ant@ant.fr	["ROLE_SOCIETE"]	\$2y\$13\$8bAYMBxWB9K9kQXC4P93xubucOs7RbNHswJmndmes2...	tak	ant
<input type="checkbox"/>	0x018e5624292d77bc88e5edf7d7174f82	lolo@lolo.fr	["ROLE_PATIENT"]	\$2y\$13\$WhKweAz9mPEBtkcFvOdtDiZ0uxSy8tYm7GD2ESXJ3...	lolo	lolo

-Les mots de passe sont hashés grâce à la création du `make:user` qui est une propriété de symfony pour créer des utilisateurs et de façon sécurisée, symfony gère automatiquement ce hashage. Le premier mot de passe est 123456 une fois hashé il ressort sous la forme `$2y$13$NldX7DjC3NE9oHq6e7gXt.dadzX83oSiKtSrWpSf9JWwdN3Ouh/6S`



Bien sûr étant donné que symfony peut modifier pour des raisons de sécurité le code de hashage la taille du champs est à 255

#### 4.5.2 La sécurisation des données insérées par l'utilisateur

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

J'ai importé la classe pour générer des UUID comme déjà précisé qui me permet d'avoir des ID générés aléatoirement et qui sont uniques encore une fois cela évite les injections SQL car lorsque les champs sont numérotés par des chiffres qui se suivent si il y a une faille, il est assez facile d'accéder à des données que l'on ne devrait pas voir.

Ensuite ne pouvant faire confiance à la saisie des utilisateurs j'ai fait deux fonctions une qui est spécialement pour les mails et l'autre pour les champs input où 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 certains 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 erreurs de majuscules et minuscules je mets tout en minuscule grâce à la fonction strtolower.

Pour mon formulaire d'inscription toujours pour des raisons d'attaque j'ai importé 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 formulaire

<pre>namespace App\Form;  use App\Entity\User; use Karser\Recaptcha3Bundle\Form\Recaptcha3Type; use Karser\Recaptcha3Bundle\Validator\Constraints\Recaptcha3; use Symfony\Component\Form\AbstractType; use Symfony\Component\Form\Extension\Core\Type\CheckboxType; use Symfony\Component\Form\Extension\Core\Type\ChoiceType; use Symfony\Component\Form\Extension\Core\Type\EmailType; use Symfony\Component\Form\Extension\Core\Type&gt;PasswordType; use Symfony\Component\Form\Extension\Core\Type\RepeatedType; use Symfony\Component\Form\Extension\Core\Type\SubmitType; use Symfony\Component\Form\Extension\Core\Type\TextType; use Symfony\Component\Form\FormBuilderInterface;  use Symfony\Component\Form\FormEvent; use Symfony\Component\Form\FormEvents; use Symfony\Component\OptionsResolver\OptionsResolver; use Symfony\Component\Validator\Constraints\IsTrue; use Symfony\Component\Validator\Constraints\Length; use Symfony\Component\Validator\Constraints\NotBlank;</pre>	<p>Ici l'on peut voir les types de champs ce qui permet d'avoir la sécurité de symfony</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------

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

```
    ]
    ->add('plainPassword', RepeatedType::class, [
        'type' => PasswordType::class,
        'first_options' => ['label' => 'Password', 'attr' => ['class' => 'form-control']],
        'second_options' => ['label' => 'Repeat Password', 'attr' => ['class' => 'form-control']],
        'mapped' => false,
        'attr' => ['autocomplete' => 'new-password'],
        'constraints' => [
            new NotBlank([
                'message' => 'Please enter a password',
            ]),
            new Length([
                'min' => 8,
                'minMessage' => 'Your password should be at least {{ limit }} characters',
                // max length allowed by Symfony for security reasons
                'max' => 4096,
            ]),
        ],
    ],
    ],
    ])
```

### 4.5.3 Sécuriser les routes

## Déploiement des projets :

### Lexique :

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

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.

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.

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

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.

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

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

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

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

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

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

**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<sup>3</sup> qui listent les tables, les champs, le détails des structures.

**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).

**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.

**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.

**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

**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

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