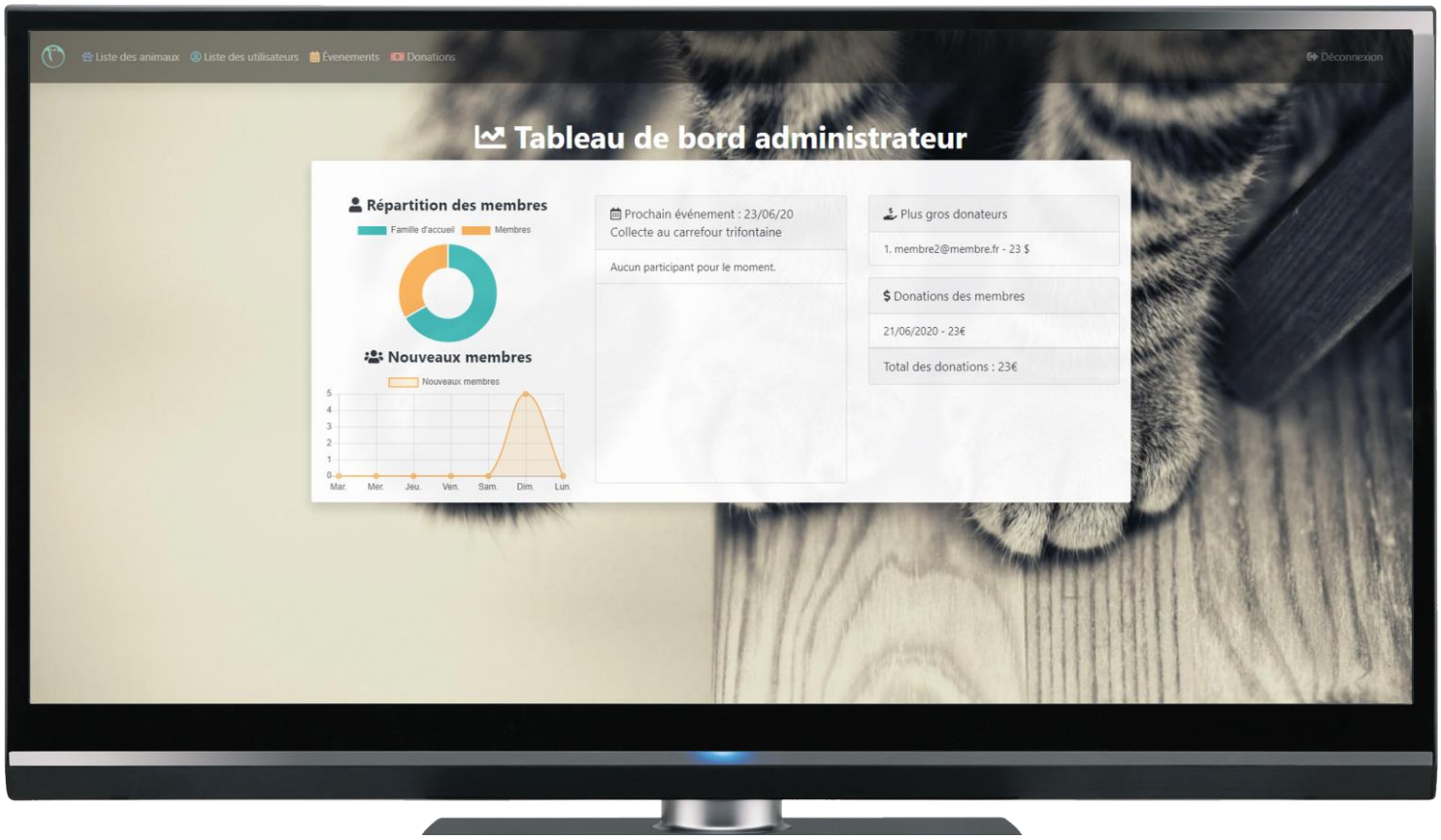
REFUGE DU LANGUEDOC

TITRE CONCEPTEUR DÉVELOPPEUR D'APPLICATIONS



REFUGE DU LANGUEDOC

Elie GAZEL
EPSI MONTPELLIER
SESSION DU 10/07/2020

TABLE DES MATIÈRES

AVANT-PROPOS

Certaines données et informations de ce rapport de projet, qu'elles soient explicites, sous-entendues ou masquées, sont strictement confidentielles. Dès lors, toute reproduction, sous quelque forme que ce soit, est formellement interdite sauf accord préalable.
Merci de votre compréhension.

INTRODUCTION

Ce projet a pour but de vous présenter les compétences techniques que j'ai acquises tout au long de ma formation. Mon entreprise n'étant pas dans le milieu du développement web, ce projet est exclusivement personnel. Ce rapport s'articule autour de trois parties.

Tout d'abord, je vous ferai une présentation de mon parcours, ainsi que de mon entreprise et du rôle que j'y occupe.

Dans un second temps, je vous présenterai le contexte m'ayant amené à effectuer ce projet, et vous présenterai tous les aspects techniques de ce dernier, de la conception à la mise en production, en passant par le développement, la mise en place de la sécurité ainsi que des tests.

Enfin, je terminerai ce dossier par un bilan personnel, où j'expliciterai mes ressentis par rapport à ce projet, et ce qu'il m'a apporté d'un point de vue technique et professionnel.

PRÉSENTATION DE L'ENTREPRISE

PRÉSENTATION DE L'ENTREPRISE

PRÉSENTATION GÉNÉRALE

« Wild Sheep Studio » est un studio de développement de jeux vidéo basé à Montpellier, fondé en 2013 par Michel Ancel, Céline Tellier, ainsi que d'autres vétérans d'Ubisoft Montpellier.

Avec un effectif d'environ une cinquantaine de personnes, le studio travaille au développement du jeu « Wild », en étroite collaboration avec Sony sous la forme d'un contrat dit « Second Party », c'est-à-dire que le produit est édité par Wild Sheep Studio, mais la propriété intellectuelle appartient à Sony.

RÔLE OCCUPÉ

Au sein de ce studio, j'occupe le rôle de Tools Programmer. Je travaille en collaboration avec de différents services, afin de fournir des outils de travail, tant aux artistes graphiques qu'aux Level Designers.

Je suis amené à développer des outils dans plusieurs langages très différents, tels que le Python, le C++, MaxScript (langage de script propriétaire du logiciel 3DSMax).

Je rencontre des problématiques intellectuellement stimulantes et, en me mettant à disposition des différents services, je suis amené à faire preuve d'une grande faculté d'adaptation et de résolution des problèmes.

LE PROJET

LE PROJET

CONTEXTE

Le cadre ayant mené au développement de ce projet est particulier.

En effet, mon entreprise se spécialisant dans un domaine très spécifique, j'ai dû signer une clause de confidentialité m'interdisant de présenter de façon précise et technique mes productions au sein de cette dernière. J'ai donc préféré développer un projet personnel, hors de l'entreprise ainsi que de ma formation.

Ma formation m'a tout de même permis de me familiariser avec de nombreux environnements de développement, langages et frameworks, qui ont influencé ma décision concernant les technologies utilisées dans ce projet.

Le but de ce projet est de répondre aux compétences demandées par le titre de **Concepteur Développeur d'Applications**, ce qui m'aurait été impossible uniquement grâce aux missions effectuées en entreprise.

Je vous présenterai donc ici une application web qui a pour vocation de rationaliser la gestion d'un refuge animalier (association de sauvetage et d'adoption), développée avec le framework PHP Symfony, intitulée « **Refuge du Languedoc** ».

Le but final de ce projet est de pouvoir le proposer à des associations afin de faciliter leur gestion et de leur donner une meilleure visibilité via une interface ergonomique et attrayante, leur permettant ainsi d'être plus efficaces dans leur mission.

PRÉSENTATION DU PROJET

Le **Refuge du Languedoc** est une application web ayant pour but de permettre à une association de sauvetage animalier une gestion complète et intuitive du cheptel d'animaux à adopter, ainsi que l'organisation de différents événements.

Cette application permettra à la personne gestionnaire de l'association de pouvoir enregistrer et présenter les animaux à l'adoption. Il aura pour ce faire une interface administrateur lui permettant de gérer également les membres de l'association, l'organisation de collectes de charité, etc.

Les membres de l'association auront accès à la liste des animaux disponibles à l'adoption, et pourront manifester leur désir d'adopter, afin de permettre au gérant de l'association d'organiser au mieux les adoptions. Ils pourront également se porter volontaires pour participer aux éventuelles collectes de charité, et faire un ou des dons pour l'association.

Les familles d'accueil pourront proposer leurs services afin d'héberger des animaux, dans l'attente de leur trouver un adoptant définitif. Ils pourront également participer aux collectes et effectuer des dons.

PRINCIPALES FONCTIONNALITÉS

- Gestion du cheptel d'animaux
- Gestion des adoptions, des hébergements selon les besoins.
- Organisation d'événements de collecte.
- Traçabilité des animaux adoptés, des dons et collectes effectués.

Ce projet est né de ma proximité en tant que famille d'accueil avec de nombreuses associations se consacrant au sauvetage animal dans la région. J'ai pu observer les nombreuses difficultés rencontrées par ces associations, souvent en sous-effectif, pour gérer leur activité. J'espère ainsi pouvoir leur offrir une solution facilitant la gestion, et leur fournir une meilleure visibilité via une interface ergonomique et attrayante, afin de leur permettre d'être plus efficace dans leur mission.

ENGLISH PROJECT PRESENTATION

The project I'm about to present is the result of an idea that came to me a while ago, when thinking about animal rescue associations.

This project is called "**Refuge du Languedoc**", its goal is to help manage a whole animal rescue association. I developed it using the PHP framework Symfony.

With this web application, the director will be able to easily manage the livestock of the association, allowing them to add new animals to the "To Adopt" list, and manage his members and foster families.

Voluntary members will have access to the list of all the animals looking for a home and show their interest in adopting one of them. They also will be able to donate to the association, or volunteer for charity collection events. Lastly, they will be able to apply to become a foster family.

Foster families will be able to provide shelter for animals, along with donating and/or volunteering for charity collection events as well. Their application to shelter an animal will be validated if they meet the criteria required for the animal (needs quarantine, is okay with dogs/cats, etc).

MAIN FEATURES

- Animal livestock management
- Fostering/Adoptions management
- Charity events organization
- Adoptions/Donations/Charity events traceability

This project is born from my experience with many animal rescue associations (as a foster family) in the region, and from the difficulties they encounter when trying to manage everything (events organization, follow up with foster families, finding families for adoptions, etc), as they often are short staffed.
My goal is to offer them a solution facilitating the management for the directors and the voluntary members and provide them with a better visibility via an ergonomic and attractive interface, helping them carry on their mission in a more efficient way.

L'ENVIRONNEMENT

L'ENVIRONNEMENT

HUMAIN

Ce projet étant produit hors du cadre de l'entreprise et de l'école, j'ai travaillé seul, en dehors de mes heures de travail.

J'avais déjà eu, lors de mes études, l'occasion de me familiariser avec le framework Symfony. C'est en me documentant plus en profondeur sur le sujet, en autodidacte, que j'ai pu obtenir les compétences nécessaires à la réalisation du dit projet.

Mes capacités d'apprentissage ainsi que la rigueur obtenue grâce à mon environnement professionnel m'ont également permis de mener à bien ce projet.

TECHNIQUE

Le point le plus important du développement du back-end de cette application est le framework **PHP** libre **Symfony**, développé par **SensioLabs**.

Ce framework suit une architecture **MVC (Modèle-Vue-Contrôleur)**. La communauté d'utilisateurs ainsi que les documentations fournies sont extrêmement efficaces, et facilitent grandement l'apprentissage. La mise à disposition de bibliothèques réutilisables est également un avantage non négligeable.

Ce sont tous ces aspects qui m'ont conduit à choisir ce framework en tant que technologie principale, et m'ont permis de développer efficacement une grande majorité des fonctionnalités de l'application.

L'**ORM Doctrine**, basé sur Doctrine DBAL (DataBase Abstraction Layer), est utilisé pour le déploiement de la base de données et pour la persistance des données.

Le front-end est développé en HTML, CSS, Twig, Javascript et jQuery.

LOGICIELS UTILISÉS :

- Visual Studio Code (IDE)
- JMerise (Conception du MCD/MLD)
- phpMyAdmin (SGBD)
- LucidChart (Création de diagrammes UML)
- Balsamiq (Maquettage)
- GitHub (Versioning), avec GitKraken (Gestion de git).
- Héroku (Déploiement)
- Composer (Gestion de dépendances)
- Cmder (Invité de commande)
- WampServer (Développement en local)
- Nuclino (Gestion de projet)

LANGAGES :

- PHP 7.2
- Symfony 4.4.2
- Doctrine 2
- HTML 5 / Twig 3.0
- CSS 3
- JS / jQuery

GESTION DE PROJET

GESTION DE PROJET

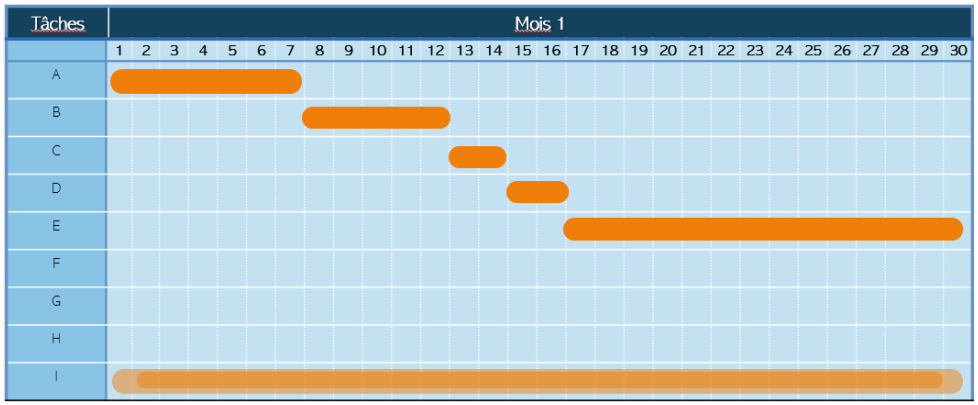
PLANIFICATION

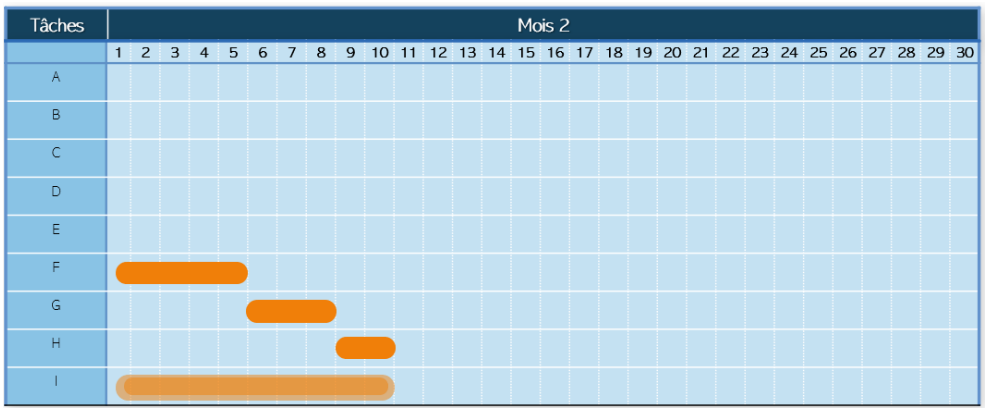
Afin d'organiser au plus tôt et de la façon la plus efficace possible mon travail, j'ai mis en place un listing des tâches sous la forme d'un diagramme de Gantt.

LISTE DES TÂCHES :

Opération	Désignation	Antériorité	Durée (j)
A	Mise en place de la gestion de projet + Premières maquettes de l'application	-	7
B	Intégration du front-end	A	5
C	Conception de la base de données	A,B	2
D	Installation de la base de données	C	1
E	Conception et Programmation Objet	D	14
F	Développement des composants	E	5
G	Mise en place des tests	F	3
H	Mise en production	G	2
I	Documentation (Tout le long du projet)	-	39

DIAGRAMME DE GANTT :

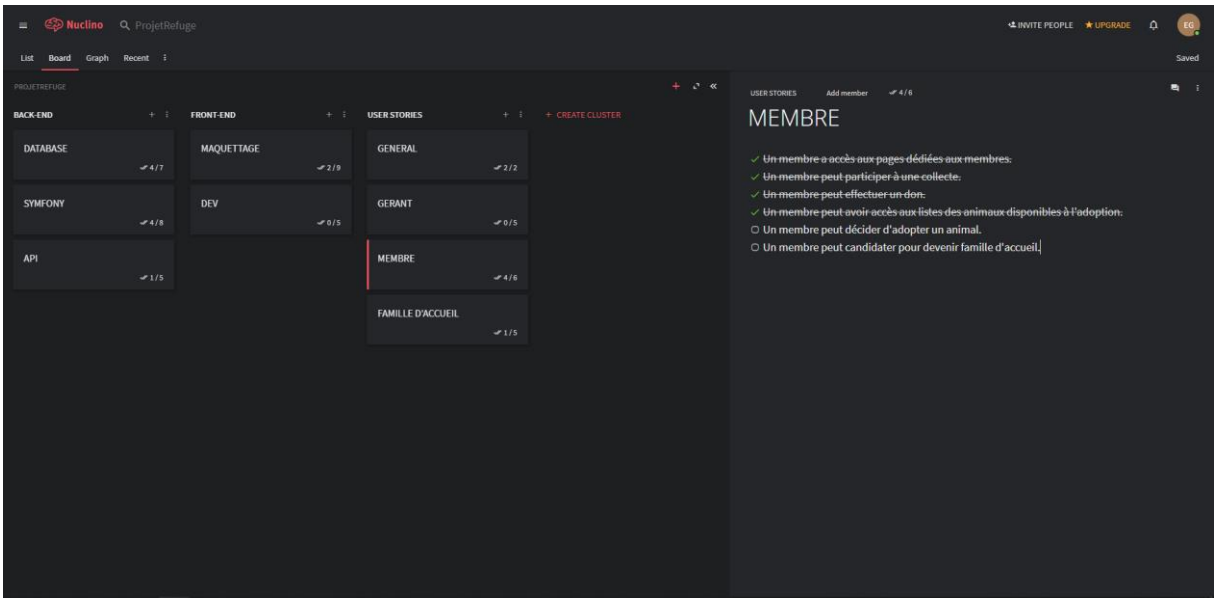


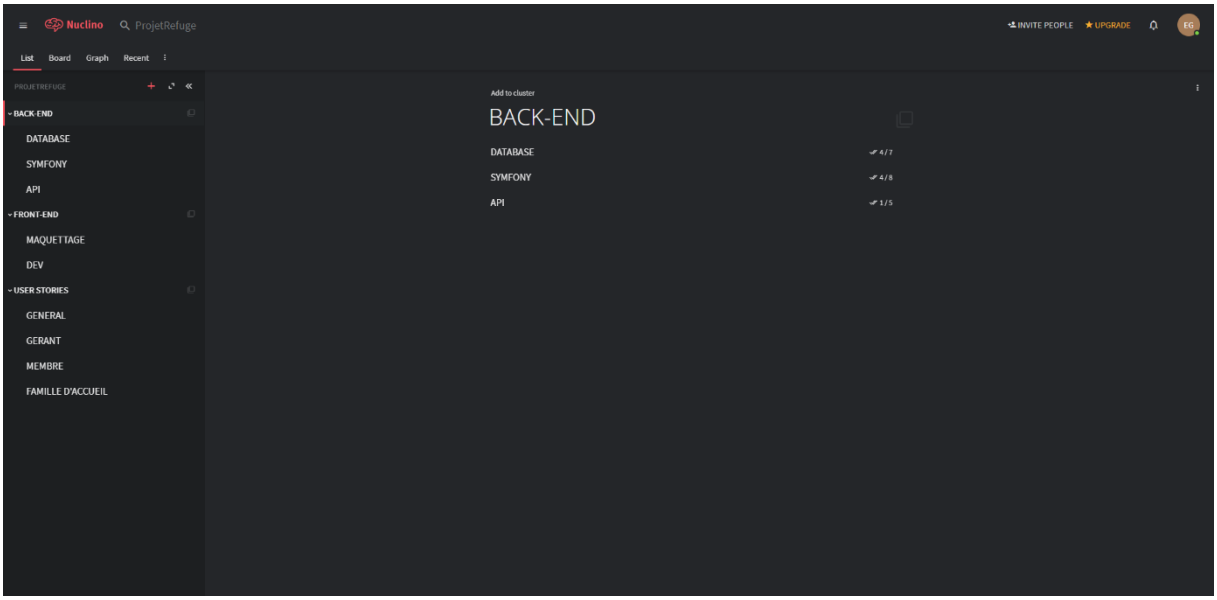


MISE EN PLACE DU NUCLINO

Afin de pouvoir suivre facilement mon avancée ainsi que les tâches restantes à effectuer, j'ai mis en place un espace de travail sur le site Nuclino.

Cette plateforme me permet de noter les différentes tâches à effectuer ainsi que leur avancée, et d'organiser le tout en clusters par catégories selon mes besoins.





MISE EN PLACE DU VERSIONING GIT

C'est tout naturellement que je me suis orienté vers le VCS GitHub, cette plateforme assurant des fonctionnalités complètes et la possibilité de créer différentes branches, nécessaires au développement de l'application.

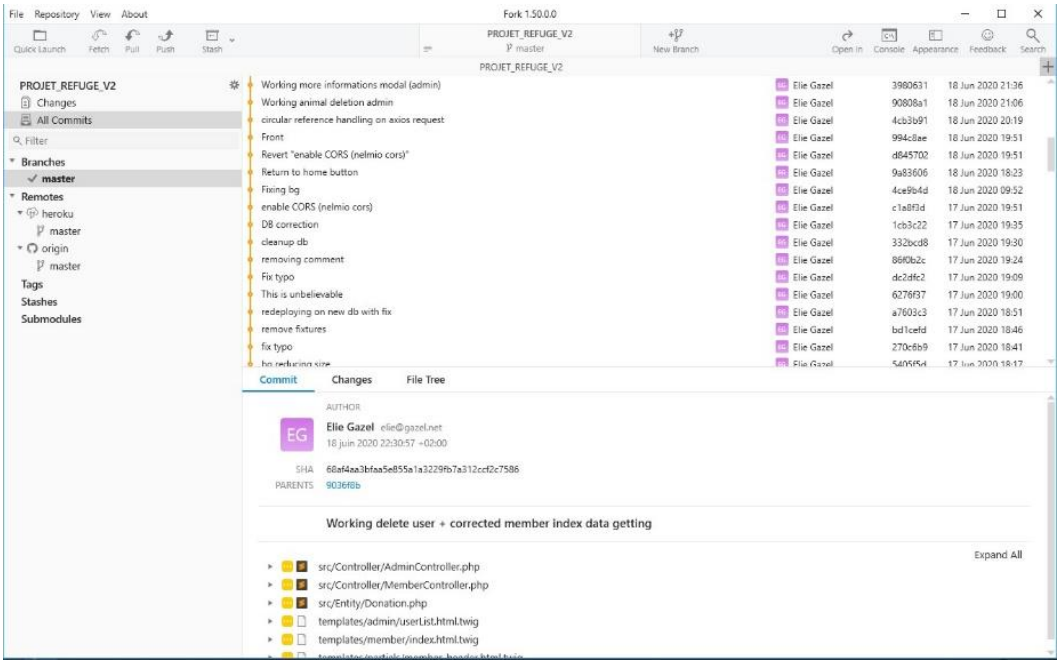
Le versioning est effectué en lignes de commandes via un terminal (Cmder), en utilisant les commandes git prévues à cet effet.

J'ai tout d'abord créé un nouveau repository sur le site de Git, puis j'ai synchronisé mon projet avec ce repository via les commandes suivantes :

- git init
- git commit -m ''first commit''
- git remote add origin https://github.com/Aotsukii/refuge_v2.git
- git push -u origin master

Chaque fonctionnalité développée l'est dans une branche nommée feature/nom-de-la-feature, et ce afin de garder une version toujours saine sur la branche Master. Les branches feature/* sont fusionnées dans la branche Master une fois la fonctionnalité terminée. Cela a pour but d'éviter les potentielles régressions.

Afin de suivre plus visuellement les branches créées et les commits effectués, j'ai utilisé le client graphique Git Fork, qui se présente comme suit :



CONCEPTION

CONCEPTION

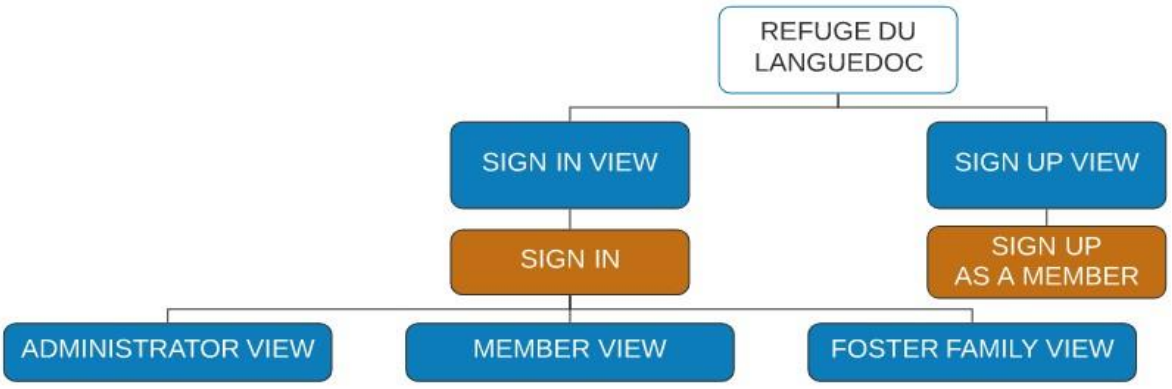
La phase de conception est nécessaire et primordiale pour le bon déroulement d'un projet, en particulier lorsqu'on développe une application.

Cette étape permet de mettre en évidence les fonctionnalités et spécificités plus facilement.

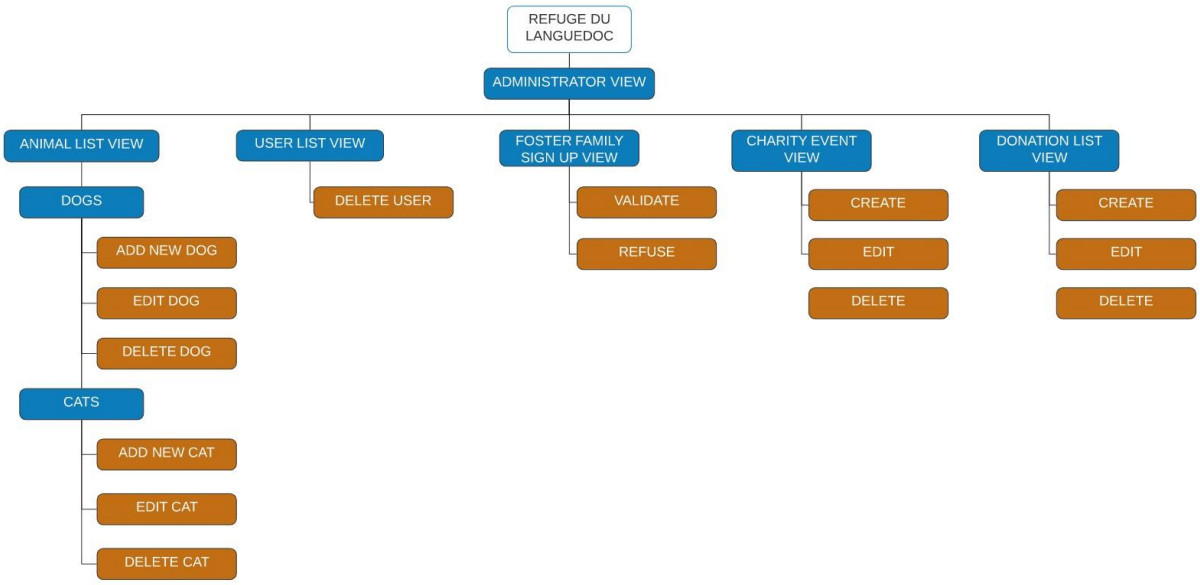
Afin de mener à bien cette étape, j'ai élaboré une arborescence complète de l'application, maquetté les vues (wireframe puis maquettes finales) et élaboré plusieurs diagrammes **UML**.

ARBORESCENCES

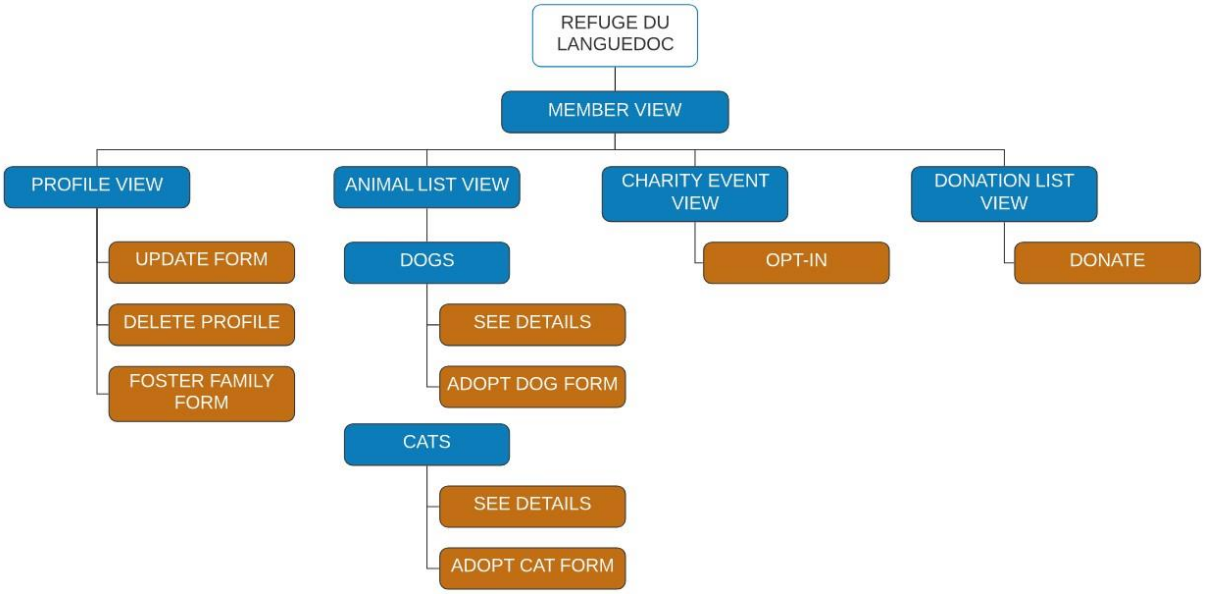
GÉNÉRALE



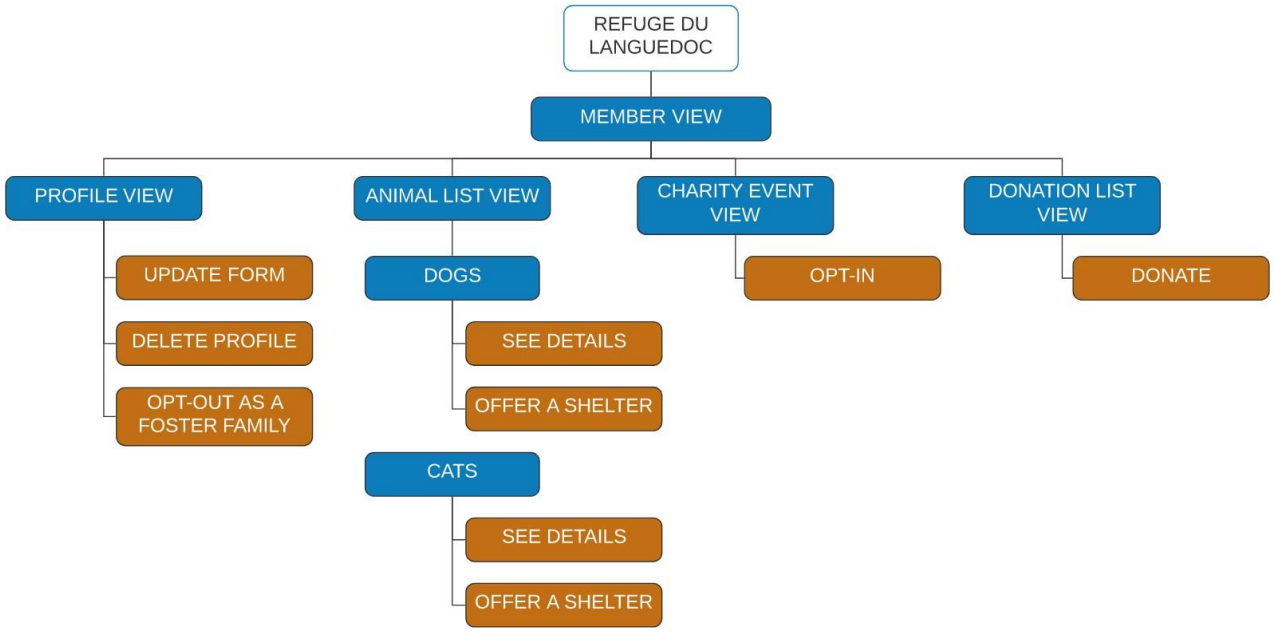
ADMINISTRATEUR



MEMBRE



FAMILLE D'ACCUEIL

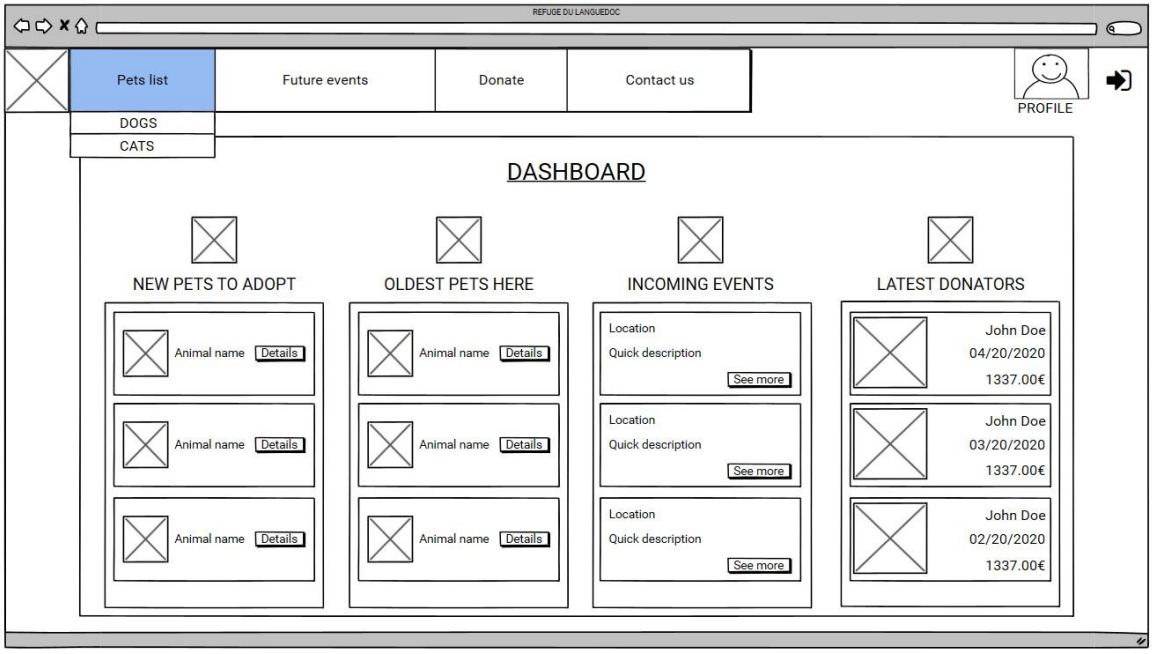


MAQUETTES

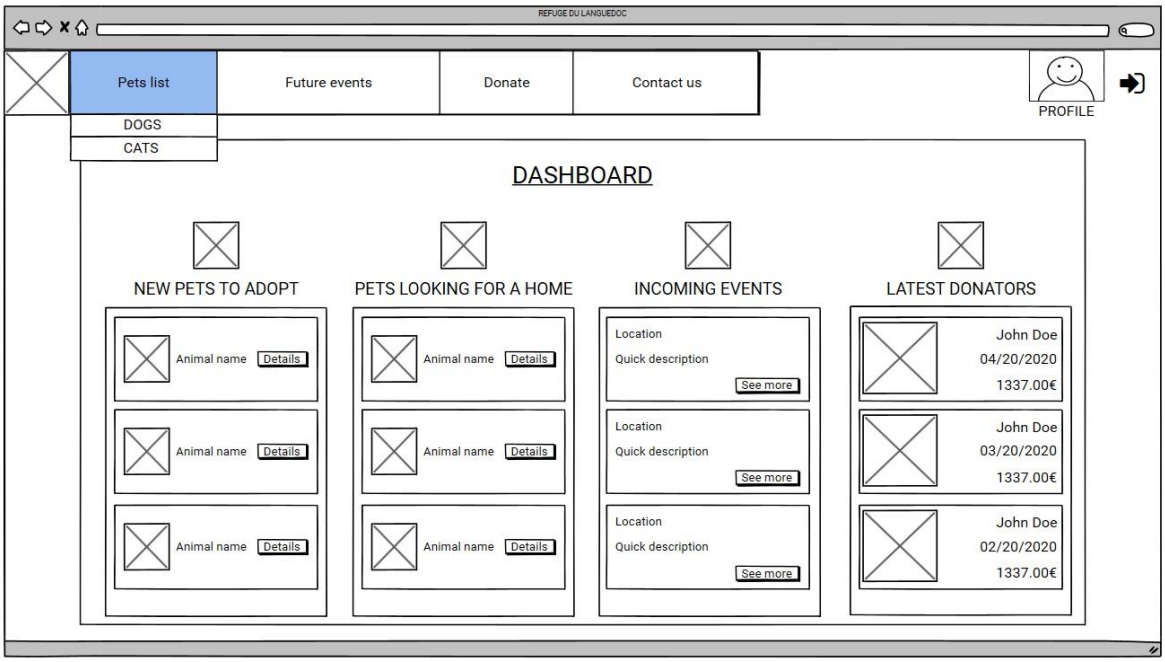
Exemples de maquettes crées avec Balsamiq.

INSCRIPTION

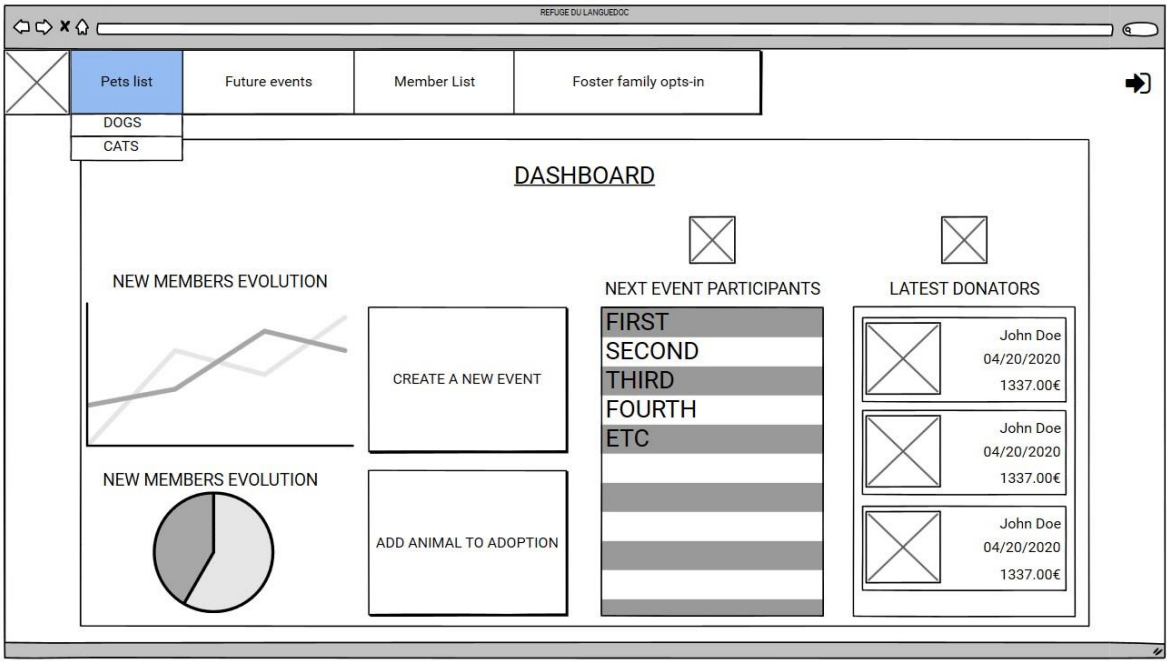
DASHBOARD MEMBRE



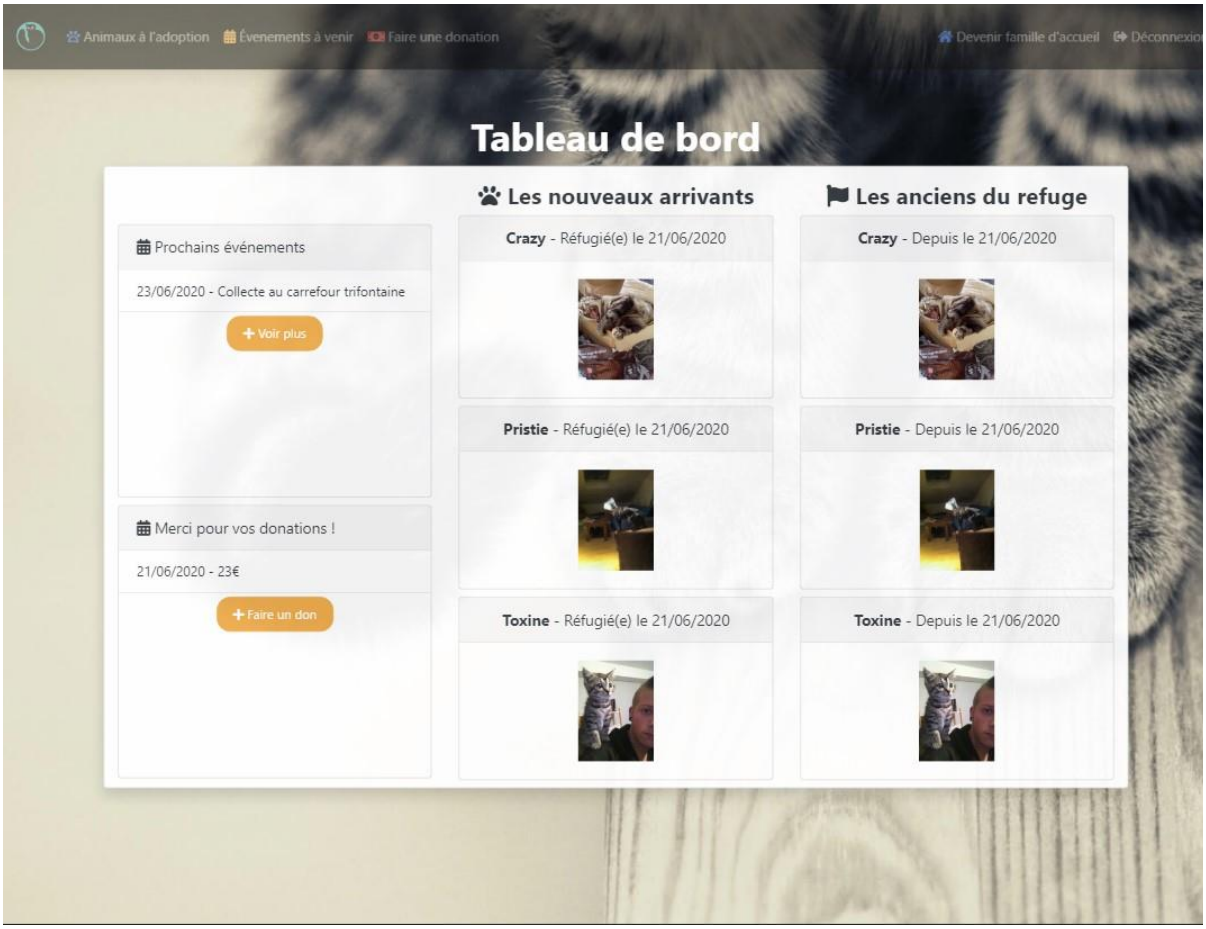
DASHBOARD FAMILLE D'ACCUEIL

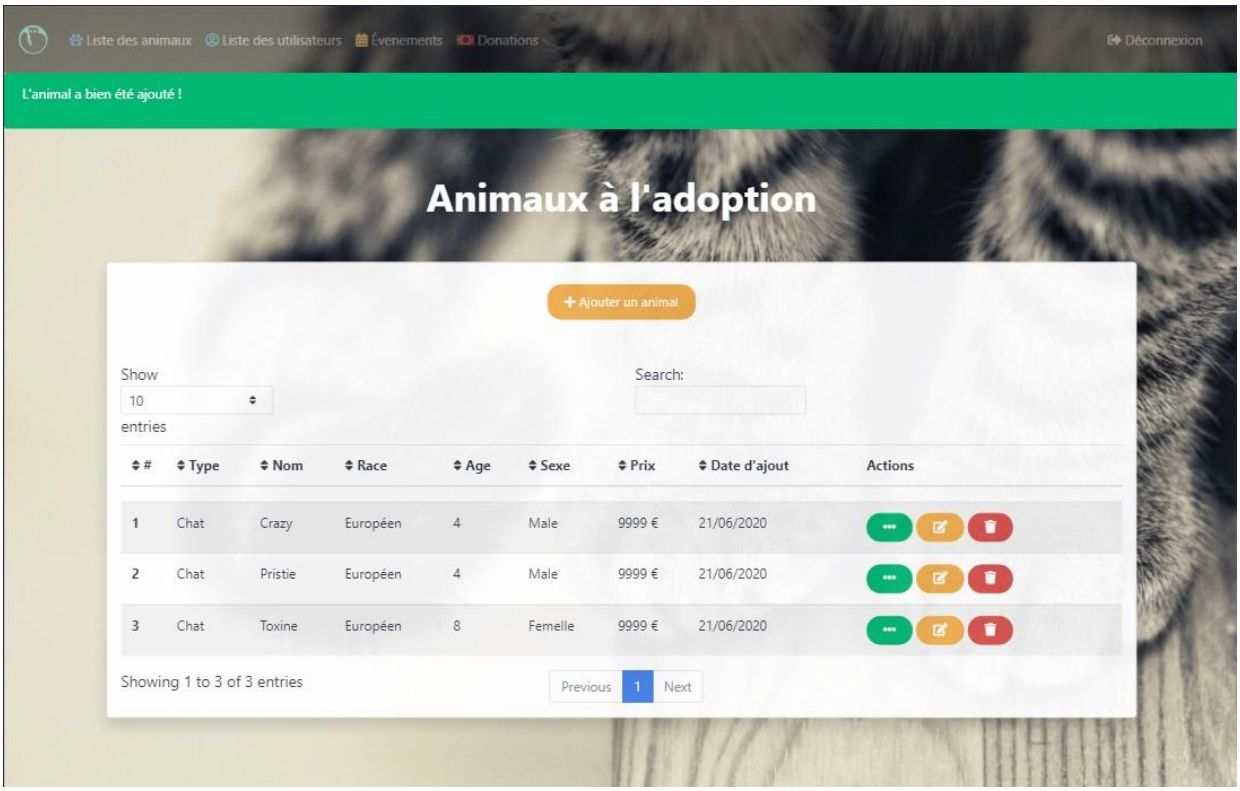


DASHBOARD ADMINISTRATEUR



MAQUETTES FINALES




Liste des animaux
Liste des utilisateurs
Évenements
Donations
Déconnexion
L'animal a bien été ajouté !
Animaux à l'adoption
+ Ajouter un animal
Show
10
entries
Search:
Type Nom Race Age Sexe Prix Date d'ajout Actions
1 Chat Crazy Européen 4 Male 9999 € 21/06/2020
2 Chat Pristie Européen 4 Male 9999 € 21/06/2020
3 Chat Toxine Européen 8 Femelle 9999 € 21/06/2020
Showing 1 to 3 of 3 entries
Previous 1 Next

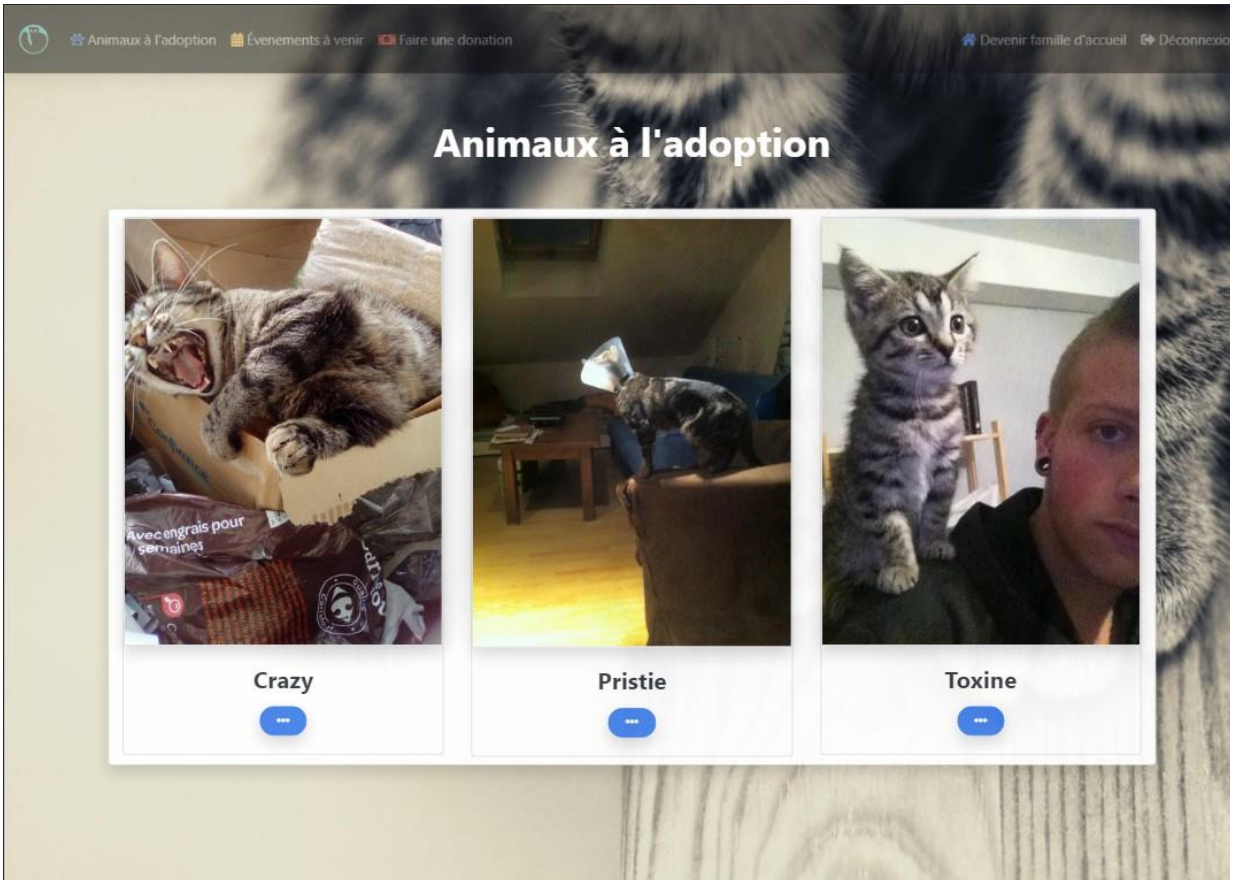

Animaux à l'adoption
Évenements à venir
Faire une donation
Devenir famille d'accueil
Déconnexio
Animaux à l'adoption
Crazy
Pristie
Toxine

DIAGRAMME DE CAS D'UTILISATION « USE CASE »

DIAGRAMME USE CASE – INSCRIPTION/CONNEXION

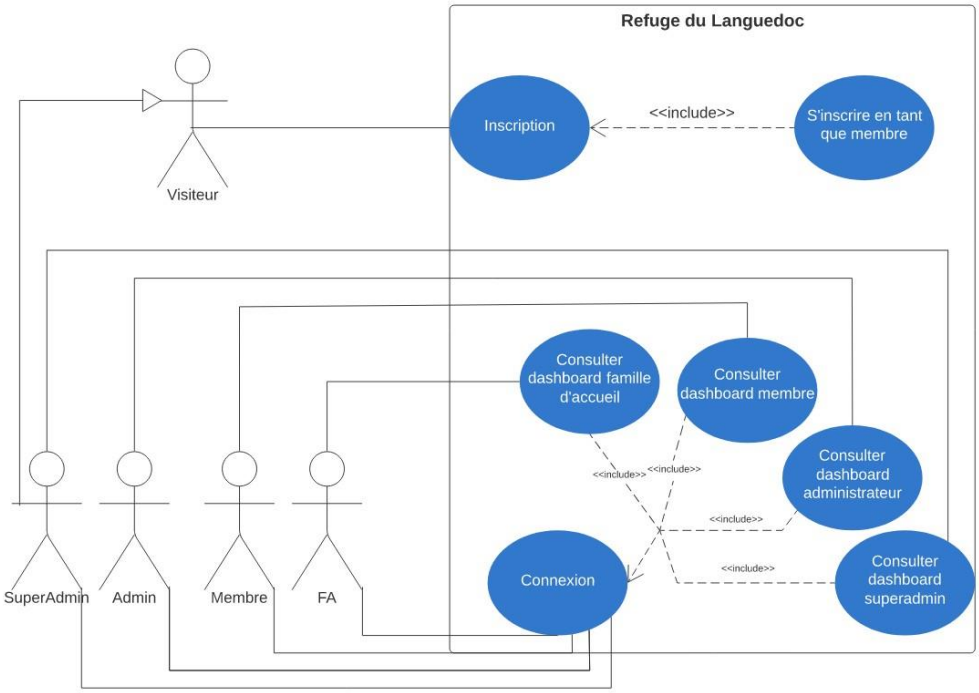


DIAGRAMME USE CASE – AJOUT ANIMAL PAR L'ADMINISTRATEUR

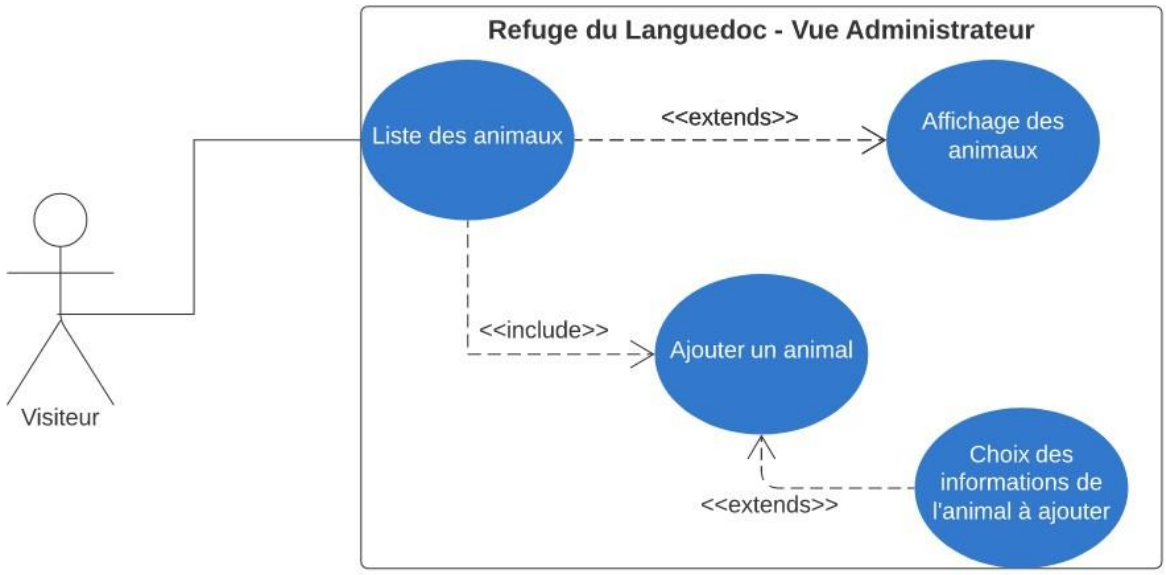
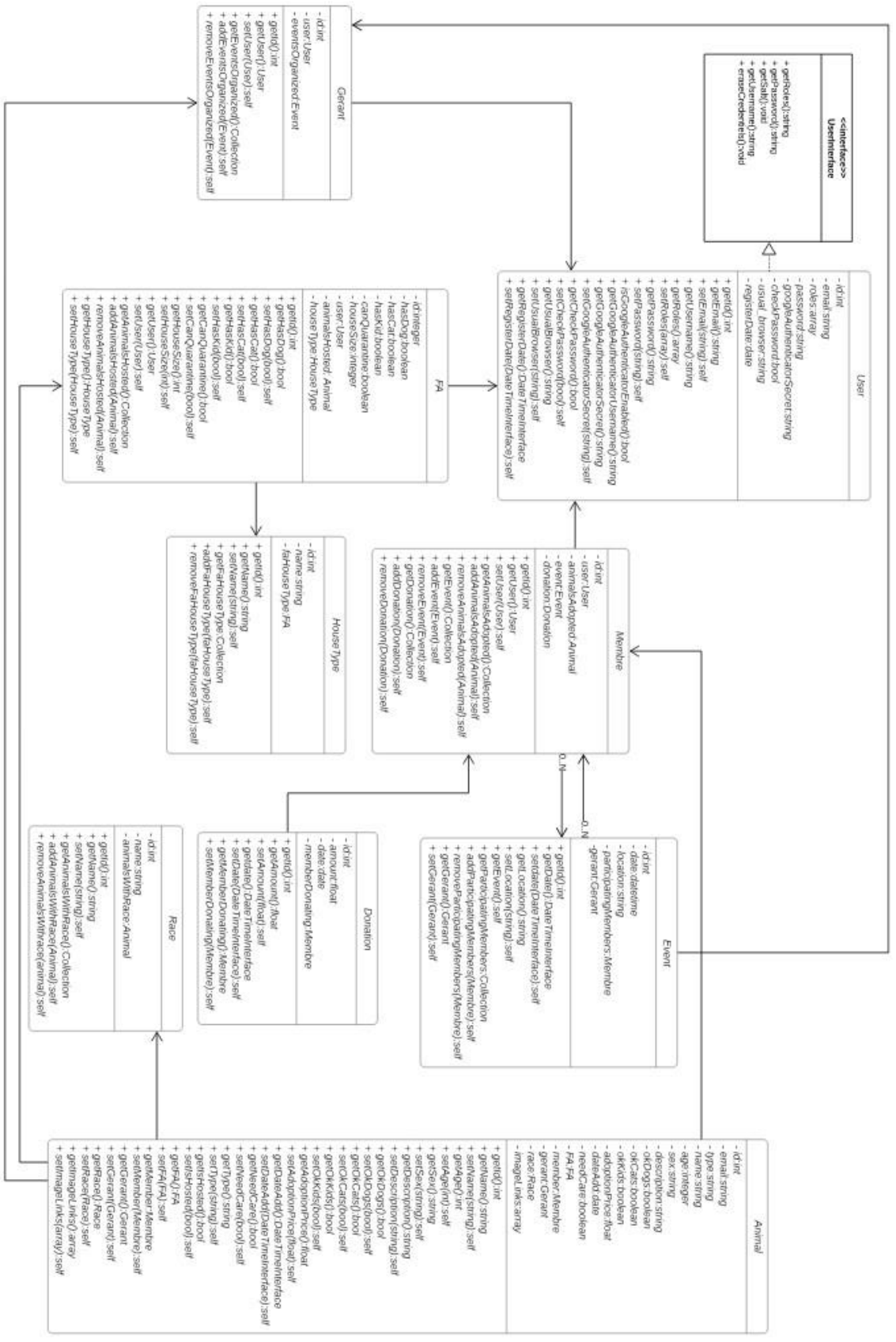


DIAGRAMME DE CLASSE DES ENTITÉS



ÉLABORATION DE LA BASE DE DONNÉES

Afin de conceptualiser la base de données, j'ai suivi la méthode Merise, qui s'organise comme suit :

- ▶ Repérage des entités.
- ▶ Etablissement des règles de gestion.
- ▶ Elaboration du dictionnaire de données.
- ▶ Définition des dépendances fonctionnelles.

REPÉRAGE DES ENTITÉS

ENTITÉS	
USER	GERANT
FA	MEMBRE
HOUSETYPE	ANIMAL
RACE	EVENT
DONATION	

RÈGLES DE GESTION

Entités liées	Description
User – Gérant	Un utilisateur correspond à un seul gérant
	Un gérant correspond à un seul utilisateur.
User – FA	Un utilisateur correspond à une seule famille d'accueil.
	Une famille d'accueil correspond à un seul utilisateur.
User – Membre	Un utilisateur correspond à un seul membre.
	Un membre correspond à un seul utilisateur.
Gérant – Animal	Un gérant peut ajouter zéro ou plusieurs animaux.
	Un animal est ajouté par un seul gérant.
Gérant – Event	Un gérant peut organiser zéro ou plusieurs événements.
	Un événement est organisé par un seul gérant.
FA – HouseType	Une famille d'accueil habite un seul type de logement.
	Un logement peut correspondre à zéro ou plusieurs familles d'accueil.

FA – Animal	Une famille d'accueil peut héberger zéro ou plusieurs animaux.
	Un animal peut être hébergé par une seule famille d'accueil.
Membre – Animal	Un membre peut adopter zéro ou plusieurs animaux.
	Un animal peut être adopté par un seul membre.
Animal – Race	Un animal a une seule race.
	Une race peut caractériser zéro ou plusieurs animaux.
Membre – Donation	Un membre peut effectuer zéro ou plusieurs donations.
	Une donation est effectuée par un seul membre.
Membre – Event	Un membre peut participer à zéro ou plusieurs événements.
	Un événement peut avoir zéro ou plusieurs membres participants.

DICTIONNAIRE DE DONNÉES

Entité	Nom donnée	Type	Taille	Remarques
User	id	int	11	auto_increment
	email	varchar	50	
	password	varchar	50	
	registerDate	datetime		
Gérant	id	int	11	auto_increment
Membre	id	int	11	auto_increment
FA	id	int	11	auto_increment
	hasDog	tinyint		
	hasCat	tinyint		
	hasKid	tinyint		
	canQuarantine	tinyint		
	houseSize	int	11	
Race	id	int	11	auto_increment
	name	varchar	50	
Donation	id	int	11	auto_increment
	montant	float		
	date	datetime		

HouseType	id	int	11	auto_increment
	libelle	varchar	255	
Event	id	int	11	auto_increment
	date	datetime		
	location	varchar	50	
Animal	id	int	11	auto_increment
	type	varchar	50	
	name	varchar	50	
	age	int	11	
	sex	varchar	50	
	description	varchar	255	
	okDogs	tinyint		
	okCats	tinyint		
	okKids	tinyint		
	adoptionPrice	float		
	dateAdd	datetime		
	needCare	tinyint		

DÉPENDANCES FONCTIONNELLES

user.id →	user.email
	user.password
	user.registerDate

gerant.id ->	**user.id**

membre.id ->	**user.id**

FA.id ->	fa.hasDog
	fa.hasCat
	fa.hasKid
	fa.canQuarantine
	fa.houseSize
	user.id
	housetype.id

race.id->	race.name

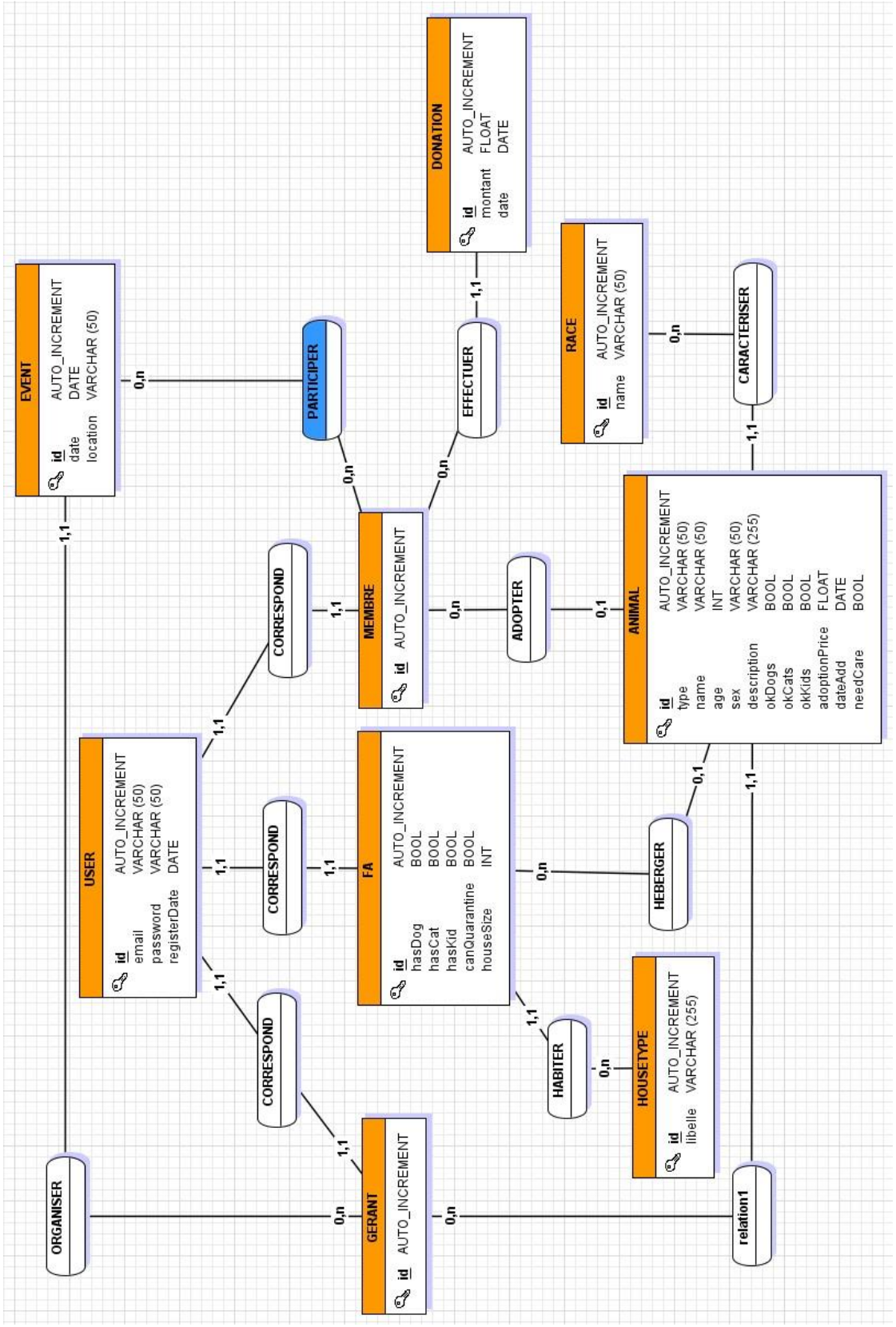
donation.id->	donation.montant
	donation.date

event.id->	event.date
	event.location
	gerant.id

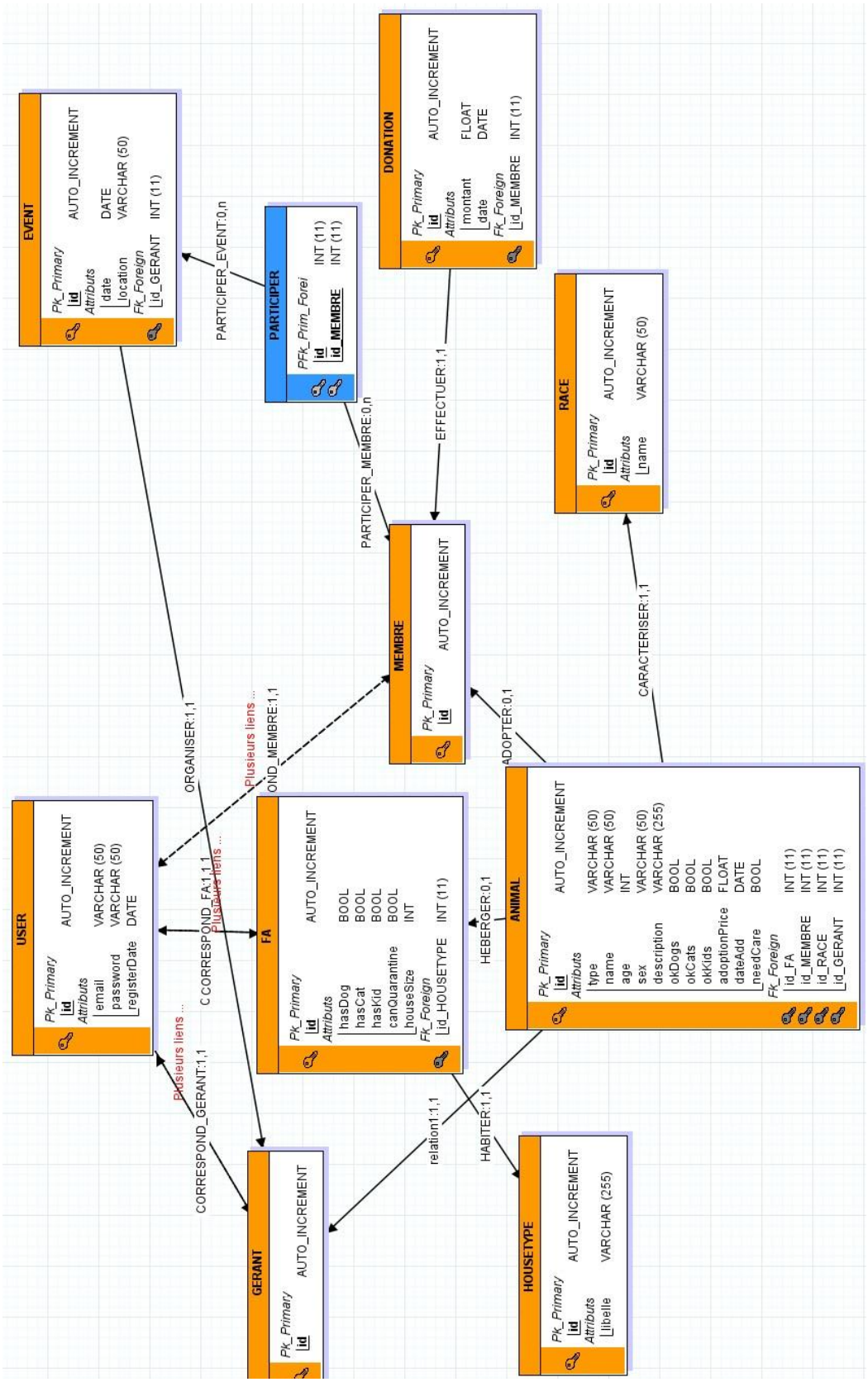
houseType.id->	housetype.libelle

animal.id ->	animal.type
	animal.name
	animal.age
	animal.sex
	animal.description
	animal.okDogs
	animal.okCats
	animal.okKids
	animal.adoptionPrice
	animal.dateAdd
	animal.needCare
	fa.id
	membre.id
	race.id
	gerant.id

MODÈLE CONCEPTUEL DES DONNÉES



MODÈLE LOGIQUE DES DONNÉS



SCRIPT SQL DE CRÉATION DES TABLES

```php
/**
 * Auto-generated Migration: Please modify to your needs!
 */
final class Version20200617173502 extends AbstractMigration
{
    public function getDescription() : string
    {
        return '';
    }

    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() !== 'mysql', 'Migration can only be executed safely on \'mysql\'.');

        $this->addSql('CREATE TABLE animal (id INT AUTO_INCREMENT NOT NULL, fa_id INT DEFAULT NULL, member_id INT DEFAULT NULL, gerant_id INT DEFAULT NULL, race_
        $this->addSql('CREATE TABLE donation (id INT AUTO_INCREMENT NOT NULL, member_donating_id INT NOT NULL, amount DOUBLE PRECISION NOT NULL, date DATE NOT NU
        $this->addSql('CREATE TABLE event (id INT AUTO_INCREMENT NOT NULL, gerant_id INT DEFAULT NULL, date DATETIME NOT NULL, location VARCHAR(255) NOT NULL, IN
        $this->addSql('CREATE TABLE fa (id INT AUTO_INCREMENT NOT NULL, user_id INT NOT NULL, house_type VARCHAR(255) DEFAULT NULL, has_dog TINYINT(1) DEFAULT NU
        $this->addSql('CREATE TABLE gerant (id INT AUTO_INCREMENT NOT NULL, user_id INT NOT NULL, UNIQUE INDEX UNIQ_D1D45C70A76ED395 (user_id), PRIMARY KEY(id))
        $this->addSql('CREATE TABLE login_attempt (id INT AUTO_INCREMENT NOT NULL, ip_address VARCHAR(50) DEFAULT NULL, date DATETIME NOT NULL COMMENT \'(DC2Type
        $this->addSql('CREATE TABLE membre (id INT AUTO_INCREMENT NOT NULL, user_id INT NOT NULL, UNIQUE INDEX UNIQ_F6B4FB29A76ED395 (user_id), PRIMARY KEY(id))
        $this->addSql('CREATE TABLE membre_event (membre_id INT NOT NULL, event_id INT NOT NULL, INDEX IDX_8D8805266A99F74A (membre_id), INDEX IDX_8D88052671F7E8
        $this->addSql('CREATE TABLE race (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf
        $this->addSql('CREATE TABLE user (id INT AUTO_INCREMENT NOT NULL, email VARCHAR(180) NOT NULL, roles JSON NOT NULL, password VARCHAR(255) NOT NULL, googl
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_6AAB231FCACAF9B4 FOREIGN KEY (fa_id) REFERENCES fa (id)');
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_6AAB231F7597D3FE FOREIGN KEY (member_id) REFERENCES membre (id)');
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_6AAB231FA500A924 FOREIGN KEY (gerant_id) REFERENCES gerant (id)');
        $this->addSql('ALTER TABLE animal ADD CONSTRAINT FK_6AAB231F6E59D40D FOREIGN KEY (race_id) REFERENCES race (id)');
        $this->addSql('ALTER TABLE donation ADD CONSTRAINT FK_31E581A0D4945564 FOREIGN KEY (member_donating_id) REFERENCES membre (id)');
        $this->addSql('ALTER TABLE event ADD CONSTRAINT FK_3BAE0AA7A500A924 FOREIGN KEY (gerant_id) REFERENCES gerant (id)');
        $this->addSql('ALTER TABLE fa ADD CONSTRAINT FK_48CB8F10A76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE gerant ADD CONSTRAINT FK_D1D45C70A76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE membre ADD CONSTRAINT FK_F6B4FB29A76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql('ALTER TABLE membre_event ADD CONSTRAINT FK_8D8805266A99F74A FOREIGN KEY (membre_id) REFERENCES membre (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE membre_event ADD CONSTRAINT FK_8D88052671F7E88B FOREIGN KEY (event_id) REFERENCES event (id) ON DELETE CASCADE');
    }

    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->abortIf($this->connection->getDatabasePlatform()->getName() !== 'mysql', 'Migration can only be executed safely on \'mysql\'.');
```

DÉVELOPPEMENT

DÉVELOPPEMENT

Le développement de l'application commence une fois la partie conception entièrement terminée.

Comme expliqué dans la partie Environnement Technique, le framework Symfony sera le cœur de l'application. Ces avantages sont multiples :

- Structuration du code (modèle MVC).
- Abstraction de la base de données (ORM).
- Gestion de la sécurité, des erreurs, des utilisateurs.
- Maintenance facile.
- Communauté très étendue et très présente.
- Mises à jour fréquentes.

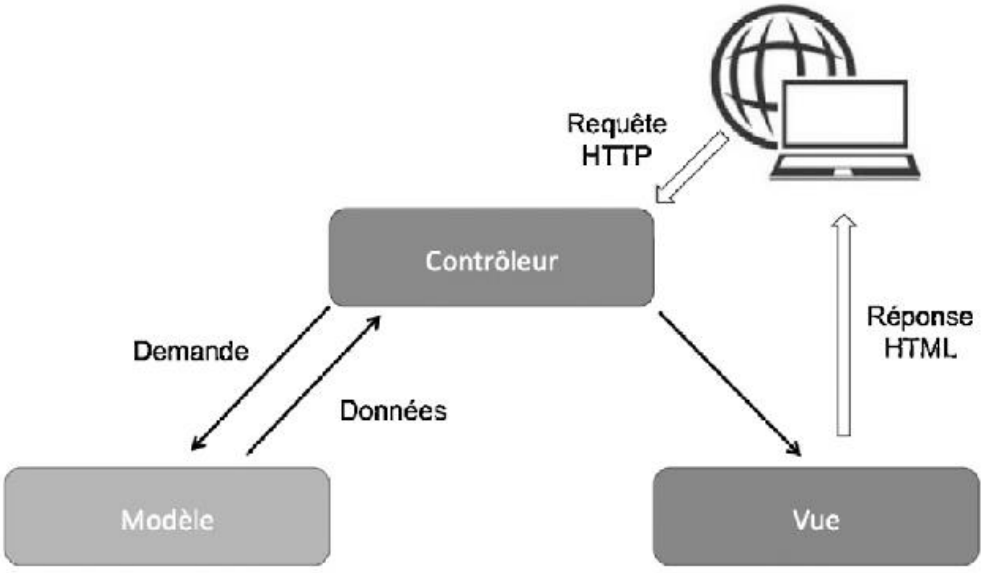
LA STRUCTURE DE DÉVELOPPEMENT

L'ARCHITECTURE MODÈLE-VUE-CONTRÔLEUR

Le modèle : Partie gérant les données du site. Va récupérer les informations brutes dans la BDD pour les transmettre au contrôleur.

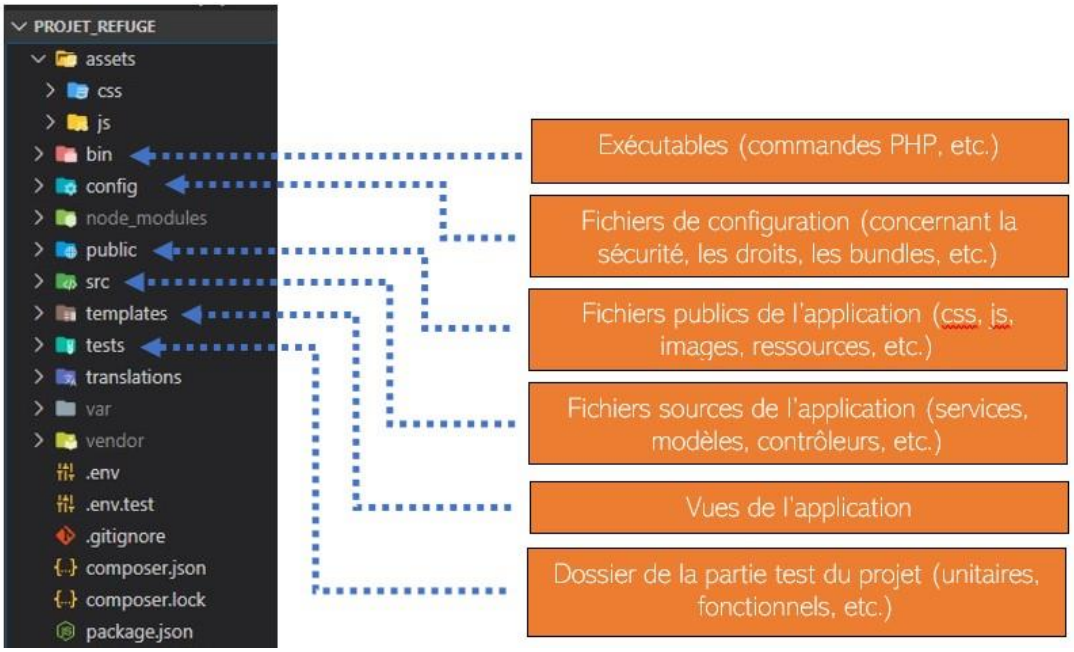
La vue : Partie gérant l'affichage. C'est ici qu'est défini ce que verra l'utilisateur.

Le contrôleur : Intermédiaire entre le modèle et la vue, en PHP. Il analyse les données et retourne à la vue le contenu à afficher.

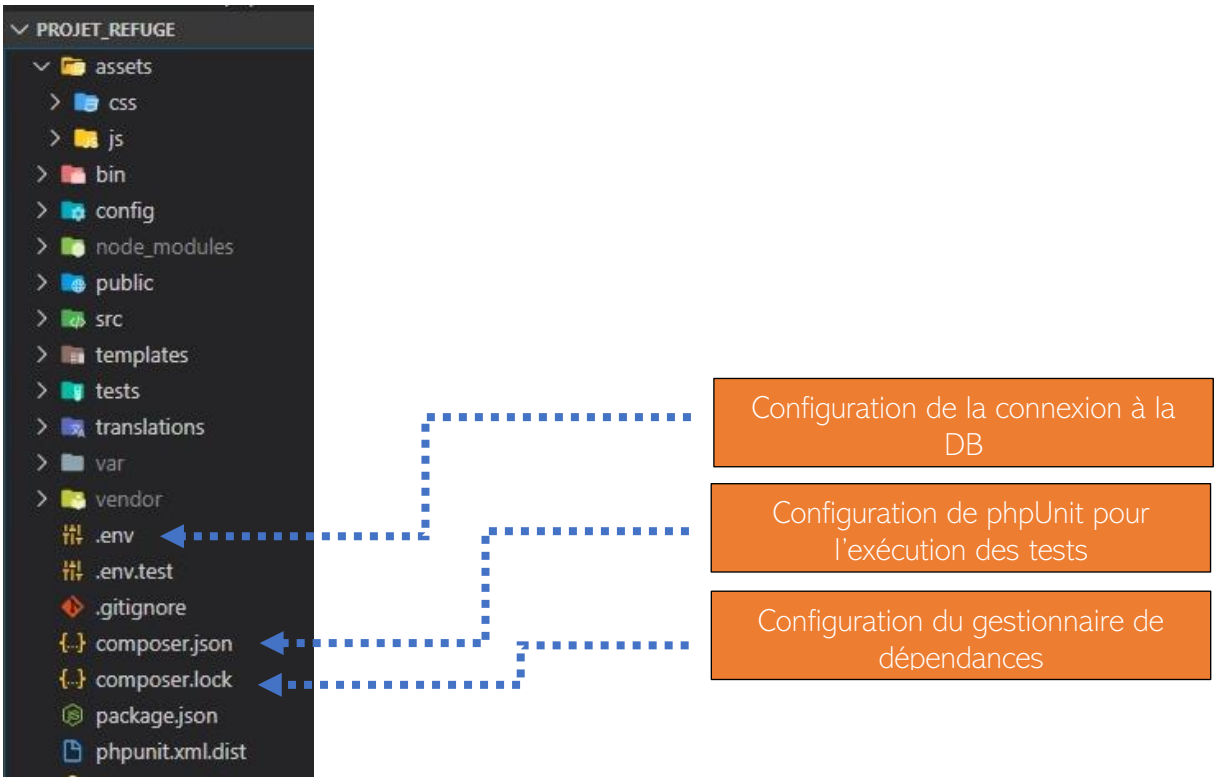


ARBORESCENCE

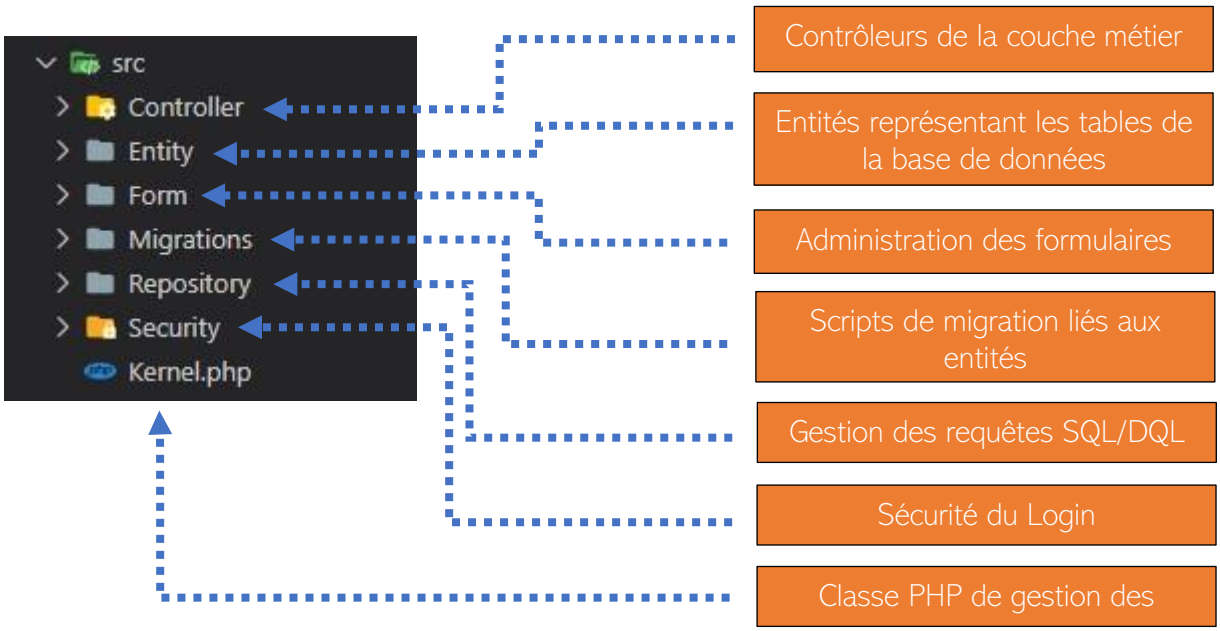
GÉNÉRALE



FICHIERS SYMFONY



DOSSIER SRC

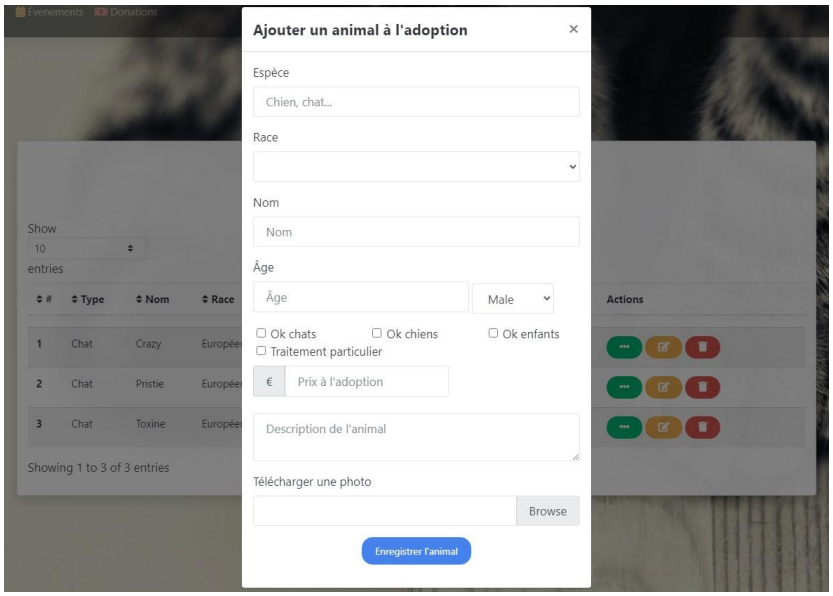

src
Controller
Entity
Form
Migrations
Repository
Security
Kernel.php
Contrôleurs de la couche métier
Entités représentant les tables de la base de données
Administration des formulaires
Scripts de migration liés aux entités
Gestion des requêtes SQL/DQL
Sécurité du Login
Classe PHP de gestion des

DÉVELOPPEMENT : MODULE GESTION DES ANIMAUX

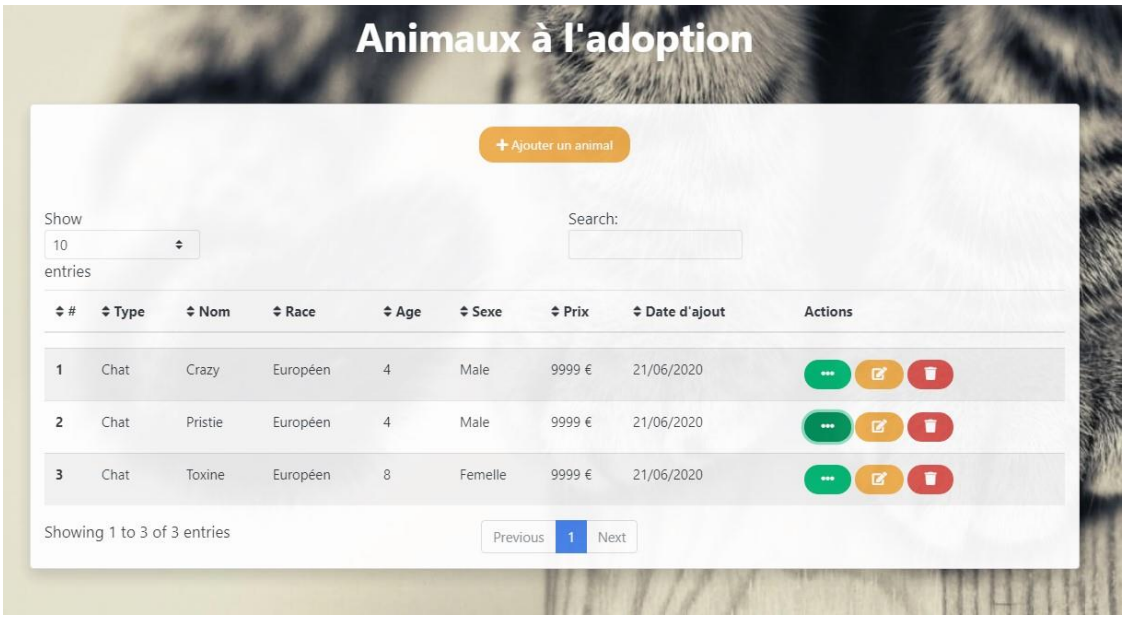
Ce module a pour but la gestion des animaux au sein du refuge, tant du côté administrateur que du côté membre et famille d'accueil.

Un administrateur a la possibilité d'ajouter/afficher/éditer/supprimer un animal.

CRÉATION D'UN ANIMAL

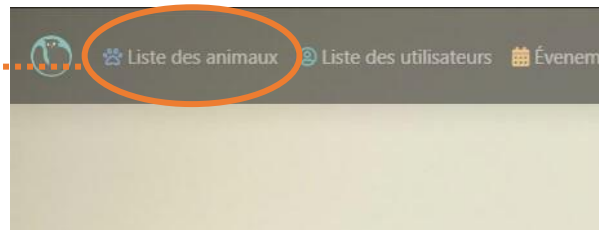


AFFICHAGE DES ANIMAUX



ACCÈS A LA VUE D'AFFICHAGE DES ANIMAUX

Requête HTTP vers la route /admin/animalList




```twig
<li class="nav-item">
    <a class="nav-link hvr-icon-pulse-grow" href="{{ path('animalList') }}">
        <i class="far fa-paw text-primary hvr-icon"></i>
        Liste des animaux
    </a>
</li>
```

Le routeur fait appel à la méthode liée à la route

```php
/**
* @IsGranted("ROLE_ADMIN")
* @Route("/admin/animalList", name="animalList")
*/
public function animalList(ContainerParametersHelpe
{
```

Le contrôleur retourne la vue désirée en fin de fonction

```php
$animals = $animalRepository->findAll();
return $this->render('admin/animalList.html.twig',
    ['animals' => $animals, 'addAnimalForm' => $addAnimalForm->createView(), 'isModalValid' => $isModal
```

Les données retournées via le contrôleur sont récupérées et affichées

```twig
{% for animal in animals %}
    {% if animals|length > 0 %}
        <tr  scope="row">
        <th>{{loop.index0 + 1}}</th>
        <td>{{animal.type}}</td>
        <td>{{animal.name}}</td>
        <td>{{animal.race}}</td>
        <td>{{animal.age}}</td>
        <td>{{animal.sex}}</td>
        <td>{{animal.adoptionPrice}} €</td>
        <td>{{animal.dateAdd.format('d/m/Y')}}</td>
        <td style="flex-direction:row;">
            <button class="btn btn-success" onClick="onClickMoreInfos({{animal.id}})" data-toggle="modal" data-target="#animalDescriptior
            <button class="btn btn-warning" data-toggle="modal" data-target="#editAnimalModalForm"><i class="fas fa-edit"></i></button>
            <button class="btn btn-danger" onClick="onClickDeleteModalConfirmationOpen({{animal.id}})" data-toggle="modal" data-target="#
        </td>
        </tr>

    {% else %}
```

AJOUT D'UN ANIMAL

L'ajout d'un animal se fait via un formulaire. Ce formulaire est géré par le composant Symfony « FormBuilder ».

Il permet une gestion dynamique et intelligente des formulaires (réutilisables, validation personnalisable, gestion des requêtes, etc).

La création d'un formulaire se fait en ligne de commande via la commande suivante : **php bin/console make:form.**

A la suite de cette commande, toujours dans notre invite de commande, on définit l'entité correspondant au formulaire généré.

Les contraintes de validation peuvent être personnalisées pour chaque champ de chaque formulaire. Ces dernières nous permettront de gérer la validité du formulaire en front-end et en back-end (grâce aux annotations sur les propriétés de l'entité), ainsi que l'affichage d'erreurs personnalisées à l'utilisateur (via le formulaire ou les annotations sur les propriétés).

L'ajout d'un animal est organisée en différentes couches. La couche métier est représentée par le contrôleur AdminController, elle permet de traiter la création du formulaire et de l'animal, ainsi que l'envoi de ce formulaire.

Couche métier : AdminController -> animalList()

```php
/**
* @IsGranted("ROLE_ADMIN")
* @Route("/admin/animalList", name="animalList")
*/
public function animalList(AnimalService $animalService, AnimalRepository $animalRepository, Request $request)
{
    $animal = new Animal();
    $addAnimalForm = $this->createForm(AnimalType::class, $animal);
    if ($addAnimalForm->handleRequest($request)){
        $animalService->createAnimal($addAnimalForm, $animal);
    }

    $animals = $animalRepository->findAll();
    return $this->render('admin/animalList.html.twig',
        ['animals' => $animals, 'addAnimalForm' => $addAnimalForm->createView(), 'isModalValid' => True]);
}
```

A
B
C
D

Légende :

A – Instanciation d'un nouvel animal.

B – Création du formulaire via le FormBuilder.

C – Test pour vérifier si la requête de création est envoyée.

D – Appel à la méthode dans le service AnimalService (Couche service).

Couche service : AnimalService -> createAnimal()

Légende :

E – Vérification de l'envoi et de la validité du formulaire

F – Persistence de l'entité crée via l'ORM.

G – Envoi de la requête en base de données via l'ORM.

H – Redirection vers la route d'affichage des animaux.

Affichage de l'animal ajouté :

Une fois l'ajout d'un animal fini, les données sont envoyées à la vue « admin/animalList.html.twig ». Cette vue est gérée de façon dynamique grâce au moteur de template **Twig**.

Les vues de l'administrateur étendent toutes un fichier nommé « admin_header.html.twig », fournissant le nécessaire pour les afficher correctement et les faire fonctionner (fichiers CSS/JS), la barre de navigation administrateur ainsi que l'espace attribué aux messages flash utilisés pour la confirmation d'une action ou sa non exécution.

Les variables envoyées par le contrôleur sont lues, parcourues et affichées grâce à ce moteur de template.

Explication du contenu du fichier « admin/animalList.html.twig »

Inclusion du fichier « admin_header.html.twig » :

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Liste des animaux à l'adoption</title>

        {% include "partials/admin_header.html.twig" %}

    </head>
    <body>
```

Utilisation de Twig pour parcourir et afficher dynamiquement les données :

```
{% for animal in animals %}
    {% if animals|length > 0 %}
        <tr  scope="row">
        <th>{{loop.index0 + 1}}</th>
        <td>{{animal.type}}</td>
        <td>{{animal.name}}</td>
        <td>{{animal.race}}</td>
        <td>{{animal.age}}</td>
        <td>{{animal.sex}}</td>
        <td>{{animal.adoptionPrice}} €</td>
        <td>{{animal.dateAdd.format('d/m/Y')}}</td>
        <td style="flex-direction:row;">
            <button class="btn btn-success" onClick="onClickMoreInfos({{animal.id}})" data-toggle="modal"
            <button class="btn btn-warning" data-toggle="modal" data-target="#editAnimalModalForm"><i clas
            <button class="btn btn-danger" onClick="onClickDeleteModalConfirmationOpen({{animal.id}})" dat
        </td>
        </tr>

    {% else %}
```

SÉCURITÉ

SÉCURITÉ

Quel que soit le type d'application développé, la mise en place d'une sécurité solide est une nécessité absolue.
Le framework Symfony permet de mettre en place une sécurité sur plusieurs parties :

- Connexion
- Connexion à la base de données
- Gestion de rôles
- Limitation d'accès aux pages
- Encodage des mots de passe
- Sécurité des formulaires
- Typage des données

DONNÉES UTILISATEURS

La CNIL est très claire à ce propos : Il est interdit de stocker en clair sur le serveur les mots de passe d'un utilisateur. Afin d'être en adéquation avec cette règle, Symfony permet la mise en place d'un encryptage avant la sauvegarde, via un algorithme d'encodage choisi parmi les nombreux disponibles dans ce framework.

config > packages > {..} security.yaml

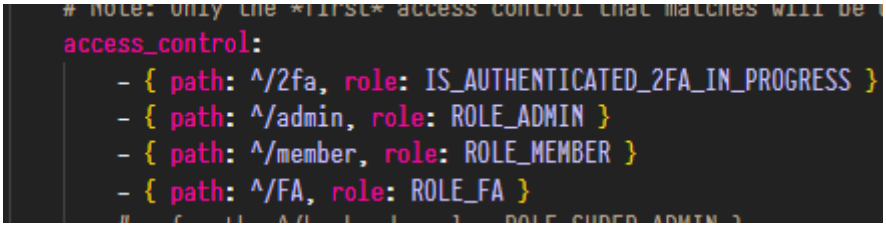
```yaml
security:
    encoders:
        App\Entity\User:
            algorithm: auto

```

Ici on voit que l'entité possédant le champ à encrypter est définie comme étant l'entité User, et l'algorithme utilisé est « auto ». C'est-à-dire que Symfony choisit l'algorithme le plus sécurisé. Cela peut changer au fil du temps, permettant une sécurité encore plus solide.

RESTRICTIONS D'ACCÈS

Afin de sécuriser les accès, j'ai défini en fonction des rôles les différentes pages et méthodes accessibles, grâce au composant « Security » proposé par Symfony, ainsi que des annotations.

Dans le fichier « /config/packages/security.yaml », je définis les rôles généraux par pattern. C'est-à-dire que toutes les routes sous la forme /admin/* ne seront accessibles qu'à l'utilisateur possédant le rôle « ROLE_ADMIN ».



Afin de m'assurer une sécurité solide au niveau des restrictions d'accès, j'ai également utilisé les annotations pour définir les droits d'accès aux méthodes dans les contrôleurs.

SÉCURITÉ DES FORMULAIRES

Symfony permet la mise en place d'une vérification d'intégrité des formulaires.

En utilisant le composant FormBuilder, nous pouvons générer pour les formulaires un jeton de sécurité CSRF (Cross-site Request Forgery) automatiquement, protégeant ainsi de l'attaque du même nom, qui consiste à injecter des données malveillantes dans des formulaires soumis par des utilisateurs légitimes.

Ce jeton de sécurité permet d' « authentifier » l'utilisateur auprès de l'application via une valeur commune. Cette vérification permet de s'assurer que toutes les données envoyées le sont par un utilisateur vérifié par ce jeton.

Exemple dans le formulaire d'authentification

```
<input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
```

Vérification de la validité du jeton lors du traitement de la demande d'authentification

```
$credentials = [
    'email' => $request->request->get('email'),
    'password' => $request->request->get('password'),
    'csrf_token' => $request->request->get('_csrf_token'),
];
```

```
public function getUser($credentials, UserProviderInterface $userProvider)
{
    $token = new CsrfToken('authenticate', $credentials['csrf_token']);
    if (!$this->csrfTokenManager->isTokenValid($token)) {
        throw new InvalidCsrfTokenException();
    }
```

DOUBLE AUTHENTIFICATION VIA JETON GOOGLE

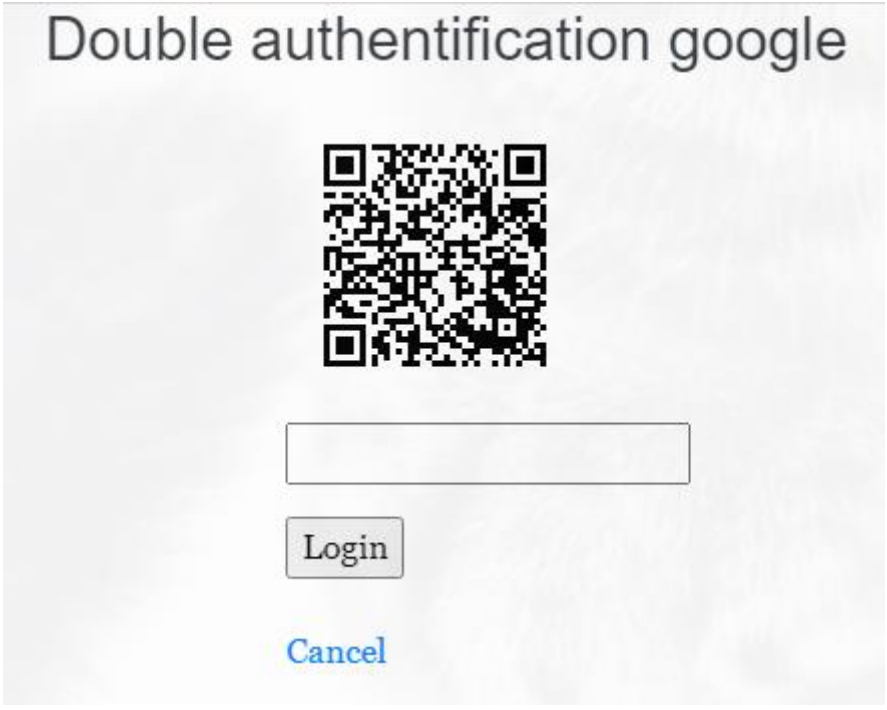
Dans le but d'éviter le vol de compte et la connexion d'une tierce personne sur un compte utilisateur, une double authentification a été mise en place, utilisant le logiciel Google Authenticator grâce au bundle « scheb/two-factor-bundle ».

Lors de la connexion, une clef est attribuée à l'utilisateur, qui génère un QR code à flasher afin d'obtenir une série de 6 chiffres servant à valider l'authentification.

Ajout de la clef secrète Google Authenticator lors de l'inscription d'un membre

```
$user->setGoogleAuthenticatorSecret($googleAuthenticatorInterface->generateSecret());
```

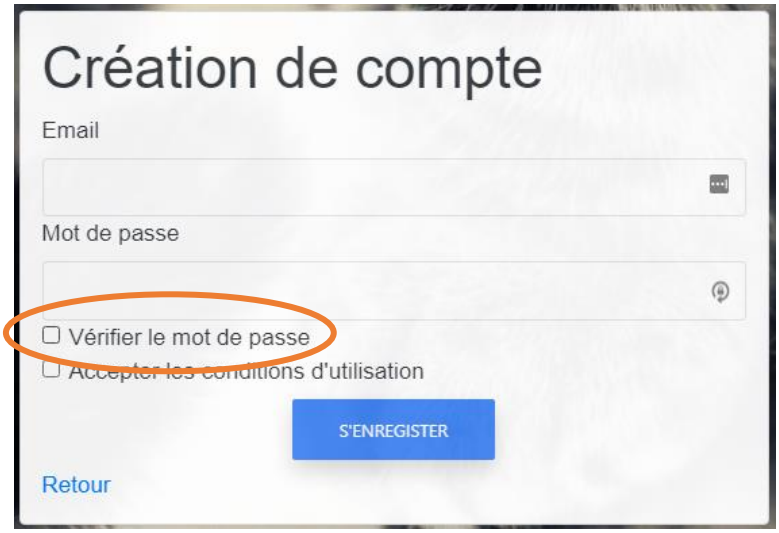
Redirection vers une route « /2fa » où l'utilisateur (ayant le rôle « IS_AUTHENTICATED_2FA_IN_PROGRESS ») devra flasher un QR code généré via sa clef secrète



Cela génèrera une clef pour le compte dans l'application smartphone Google Authenticator. Entrer cette clef permettra de finaliser la connexion et la redirection vers le site selon le rôle de l'utilisateur.

VÉRIFICATION DU MOT DE PASSE À L'INSCRIPTION

Afin de m'assurer que les utilisateurs, lors de leur inscription, utilisent un mot de passe sécurisé, je fais appel à un validateur Symfony spécifique : « Compromised Password Validator » Cette option n'est pas obligatoire lors de l'inscription :



Ce validateur permet de vérifier le mot de passe en le comparant aux listes de comptes fuités lors des failles de sécurité les plus connues (sur des gros sites, etc), en faisant un appel à l'API « HaveIBeenPwned ».

Dans le cas où cette option est utilisée lors de l'inscription, si le mot de passe est trouvé dans les bases de données des comptes fuités, une erreur signifiera à l'utilisateur que son mot de passe n'est plus sécurisé.

Mise en place du validateur dans la procédure d'inscription

```php
// If compromised assign the error to the password field
if ($user->getCheckPassword() && $violations instanceof ConstraintViolationList && $violations->count()) {
    $password = $form->get('plainPassword');
    if ($password instanceof Form) {
        $violationMessage = "Ce mot de passe a été divulgué lors d'une violation de données, il ne doit pas être
        $password->addError(new FormError((string) $violationMessage));
    }
} else {
    // encode the plain password
```

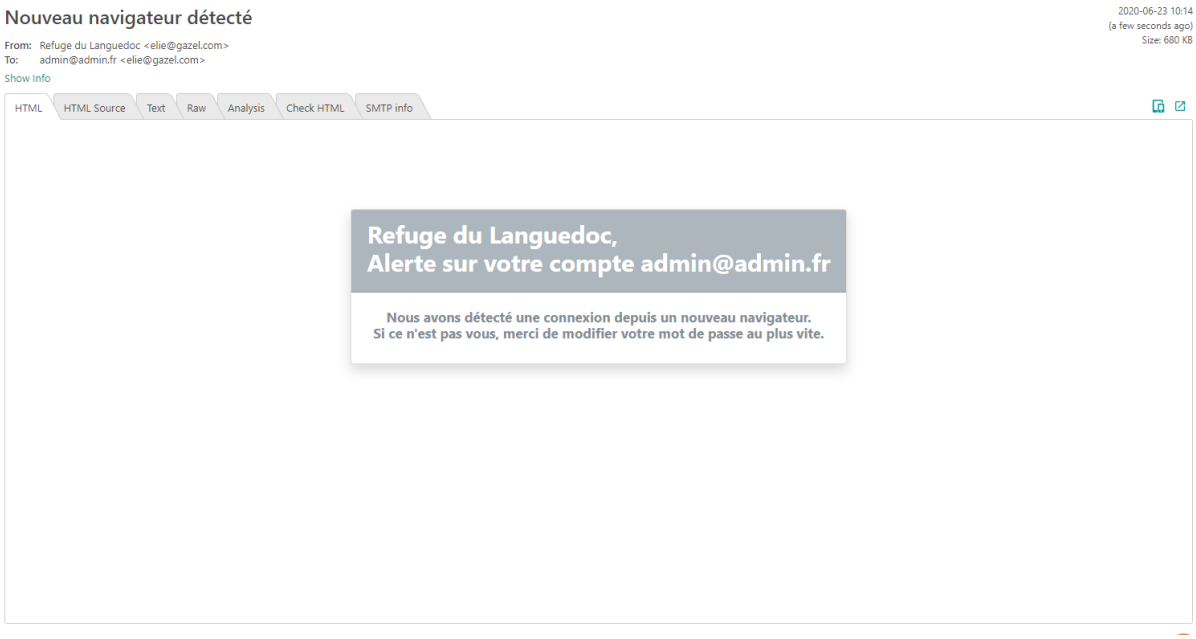
NOTIFICATION DE CHANGEMENT DE NAVIGATEUR

Dans une démarche de sécurisation maximale des comptes des utilisateurs et donc de l'application en elle-même, j'ai mis en place une vérification, lors de chaque connexion, du navigateur utilisé par l'utilisateur.

Si le navigateur utilisé au moment de la connexion n'est pas le même que celui utilisé lors de la dernière connexion, un envoi de mail de notification est déclenché. Cela ne bloque pas la connexion.

```php
$savedBrowserForCurrentUser = $currentUser->getUsualBrowser();

$u_agent = $_SERVER['HTTP_USER_AGENT'];
$bname = 'Unknown';

// Next get the name of the useragent yes seperately and for good reason
if(preg_match('/MSIE/i',$u_agent) && !preg_match('/Opera/i',$u_agent)){
  $bname = 'Internet Explorer';
}elseif(preg_match('/Firefox/i',$u_agent)){
  $bname = 'Mozilla Firefox';
}elseif(preg_match('/OPR/i',$u_agent)){
  $bname = 'Opera';
}elseif(preg_match('/Chrome/i',$u_agent) && !preg_match('/Edge/i',$u_agent)){
  $bname = 'Google Chrome';
}elseif(preg_match('/Safari/i',$u_agent) && !preg_match('/Edge/i',$u_agent)){
  $bname = 'Apple Safari';
}elseif(preg_match('/Netscape/i',$u_agent)){
  $bname = 'Netscape';
}elseif(preg_match('/Edge/i',$u_agent)){
  $bname = 'Edge';
}elseif(preg_match('/Trident/i',$u_agent)){
  $bname = 'Internet Explorer';
}

if ($bname ≠ $savedBrowserForCurrentUser) {
    if ($savedBrowserForCurrentUser≠NULL){

        $email = (new TemplatedEmail())->from(new Address('elie@gazel.com', 'Refuge du Languedoc'))
                                       ->to(new Address('elie@gazel.com', $currentUser->getUsername()))
                                       ->subject('Nouveau navigateur détecté')
                                       ->htmlTemplate('email/browser.html.twig')
                                       ->context([
                                           'user' => $currentUser
                                       ]);
        $this->mailer->send($email);
        $currentUser->setUsualBrowser($bname);

    } else {
        $currentUser->setUsualBrowser($bname);
    }
}
```

Mail reçu lors d'une connexion via un nouveau navigateur



Template du mail « /templates/email/browser.html.twig »

```
<body>
    <div class="container">
        <div class="row d-flex justify-content-center" style="margin-top: 10%">
            <div class="card shadow">
                <div class="card-header bg-secondary">
                    <h3 class="text-white">
                        Refuge du Languedoc, <br>
                        Alerte sur votre compte {{ user.username }}
                    </h3>
                </div>
                <div class="card-body">
                    <h6 class="text-center text-muted">Nous avons détecté une connexion depuis un nouveau navigateur.<br>
                    Si ce n'est pas vous, merci de modifier votre mot de passe au plus vite.</h6>
                </div>
            </div>
        </div>
    </div>
</body>
```

BLOQUAGE DES CONNEXIONS SUCCESSIVES

Enfin, afin de rendre impossible toute tentative d'attaque par force brute (connexion à un compte via l'utilisation d'un dictionnaire de mots de passe par exemple), une historisation des connexions a été mise en place.
Si le système perçoit plus de 3 connexions successives dans un laps de temps court (3 minutes) sur le même compte, un bloquage de cinq minutes est mis en route.

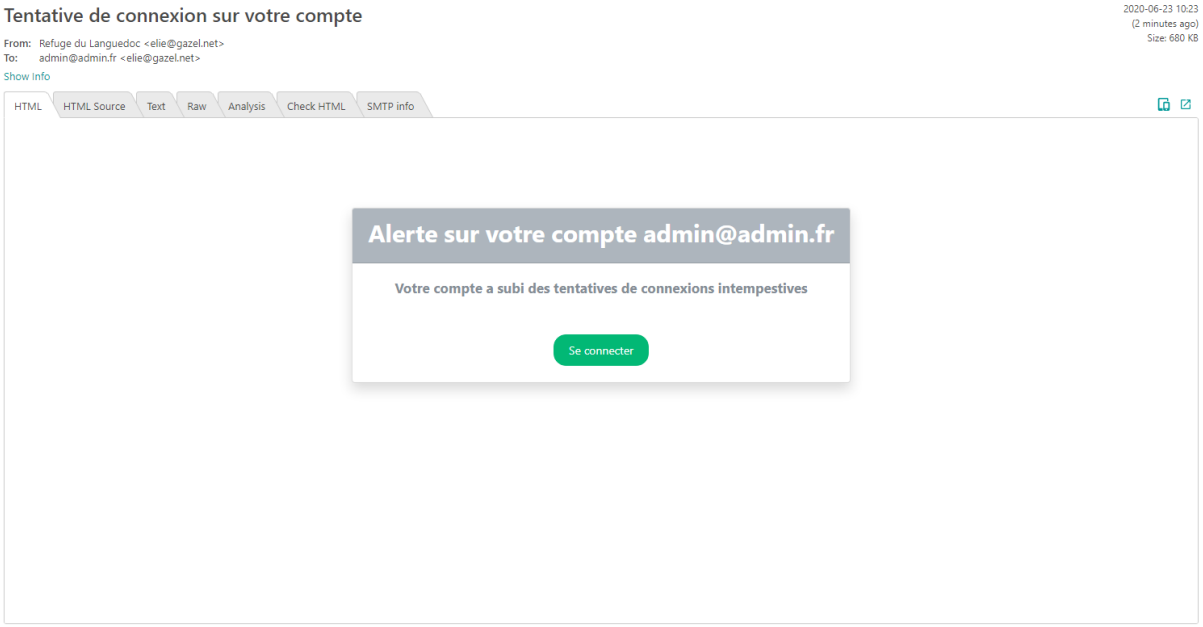
Requête DQL utilisée pour compter les tentatives récentes de connexion

```php
const DELAY_IN_MINUTES = 3;

public function __construct(ManagerRegistry $registry)
{
    parent::__construct($registry, LoginAttempt::class);
}

public function countRecentLoginAttempts(string $username): int
{
    $timeAgo = new \DateTimeImmutable(sprintf('-%d minutes', self::DELAY_IN_MINUTES));

    return $this->createQueryBuilder('la')
        ->select('COUNT(la)')
        ->where('la.date >= :date')
        ->andWhere('la.username = :username')
        ->getQuery()
        ->setParameters([
            'date' => $timeAgo,
            'username' => $username,
        ])
        ->getSingleScalarResult()
    ;
}
```

Implémentation lors de la vérification des identifiants

```php
if ($this->loginAttemptRepository->countRecentLoginAttempts($credentials['email']) > 3) {
    $email = (new TemplatedEmail())->from(new Address('elie@gazel.net', 'Refuge du Languedoc'))
                                   ->to(new Address('elie@gazel.net', $user->getUsername()))
                                   ->subject('Tentative de connexion sur votre compte')
                                   ->htmlTemplate('email/security.html.twig')
                                   ->context([
                                       'user' => $user
                                   ]);

    $this->mailer->send($email);
    throw new CustomUserMessageAuthenticationException('Trop de tentatives de connexion d\'affilée. Veuillez patienter avant de re-essayer');
}
```

Mail reçu dans le cas où trop de tentatives d'affilée sont enregistrées



Template de mail « /templates/email/security.html.twig »

```twig
<body>
    <div class="container">
        <div class="row d-flex justify-content-center" style="margin-top: 10%">
            <div class="card shadow">
                <div class="card-header bg-secondary">
                    <h3 class="text-white">
                        Alerte sur votre compte {{ user.username }}
                    </h3>
                </div>
                <div class="card-body">
                    <h6 class="text-center text-muted">Votre compte a subi des tentatives de connexions intempestives</h6>
                </div>
                <div class="card-body d-flex justify-content-center">
                    <a href="{{ url('app_login') }}" class="btn btn-success">Se connecter</a>
                </div>
            </div>
        </div>
    </div>
</body>
```

BASE DE DONNÉES

L'accès à la base de données doit être extrêmement sécurisée, car c'est le lieu de regroupement de toutes les informations importantes de l'application.

Il faut créer un utilisateur spécifique, ayant accès uniquement à la base de données de l'application, et aucune autre. Cet utilisateur ne pourra pas créer d'autres comptes utilisateurs, ni modifier les permissions depuis la base de données avec ce compte.

Commande de création d'un compte utilisateur MySQL

```
>   CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
```

Edition des privilèges du compte créé

```
>GRANT ALL PRIVILEGES ON db_refuge . * TO 'elie.gazel'@'localhost'

REVOKE CREATE, UPDATE PRIVILEGES ON db_refuge.user TO 'elie.gazel'@'localhost'
```

TESTS

TESTS

INTRODUCTION

Afin d'éviter au maximum les potentielles régressions, des tests unitaires et fonctionnels ont été implémentés au sein de l'application.

Ces tests me permettent d'identifier un maximum de comportements problématiques, pouvant nuire à l'expérience d'utilisation de l'application ainsi qu'à son bon fonctionnement.

Les tests unitaires ont pour but de vérifier le bon fonctionnement d'une fonction spécifique, tandis que les tests fonctionnels vérifient le bon fonctionnement de l'application en général dans le cadre d'une utilisation réelle.

Ces tests sont effectués en « boîte noire », ce qui signifie que l'on n'a aucune information sur le fonctionnement du code en interne. Ces tests sont donc non intrusifs, et vérifient que pour une donnée spécifique en entrée, on obtient bien le résultat souhaité en retour sans rentrer dans le détail du code testé.

Lors du lancement de la procédure exécutant les tests en ligne de commande, la validation des tests se présente comme suit :

Ces tests sont également effectués lors du déploiement automatisé de l'application et sont bloquants, c'est-à-dire qu'un test retournant une erreur bloquera le déploiement, dans le but d'éviter au maximum les potentielles régressions.

EXEMPLE DE TEST FONCTIONNEL

```php
class IndexControllerTest extends WebTestCase
{
    public function testIndex()
    {
        // create a client : mimicking someone browsing the website.
        $client = static::createClient();

        // request access to the route '/'
        $client->request('GET', '/');

        // Assert that the result returned is code 200, meaning the page is up.
        $this->assertEquals(200, $client->getResponse()->getStatusCode());
    }
}
```

Ce test fonctionnel a pour but de vérifier que la page d'accueil est bien accessible, et que lorsqu'on effectue une requête sur la route, l'accès se fait sans problème.

```php
$client = static::createClient();
```

On créé un client, représentant un utilisateur naviguant sur le site.

```php
$client->request('GET', '/');
```

On effectue une requête GET sur la route « / », correspondant à l'accès à la page par l'utilisateur.

```php
$this->assertEquals(200, $client->getResponse()->getStatusCode());
```

On vérifie que la réponse retournée par la requête correspond au code HTTP 200 (signifiant la réussite d'une requête, et donc la disponibilité de la page d'accueil).

EXEMPLES DE TESTS UNITAIRES

```php
class UserTest
{
    public function testGetRoles(){
        $this->user->setRoles('ROLE_ADMIN');
        $this->assertEquals("[ROLE_ADMIN]", $this->user->getRoles());
    }

    public function testGetEmail(){
        $this->user->setEmail('elie@gazel.net');
        $this->assertEquals("elie@gazel.net", $this->user->getEmail());
    }
}
```

Ici nous avons deux tests unitaires, vérifiant le bon fonctionnement des getter/setter de l'entité User.

```php
$this->user->setRoles('ROLE_ADMIN');
```

On utilise le setter pour attribuer à l'utilisateur le rôle « ROLE_ADMIN »

```php
$this->assertEquals("[ROLE_ADMIN]", $this->user->getRoles());
```

On vérifie que le résultat retourné par le getter correspond bien au résultat attendu (Ici un tableau comportant le rôle précédemment attribué).

```php
$this->user->setEmail('elie@gazel.net');
```

Même procédure, on utilise le setter pour définir une adresse mail à l'utilisateur.

```php
$this->assertEquals("elie@gazel.net", $this->user->getEmail());
```

Et on vérifie que le résultat retourné par le getter correspond à l'adresse attribuée au dessus.

MISE EN PRODUCTION

MISE EN PRODUCTION

INTRODUCTION

La mise en production est l'étape consistant à la migration du code de développement en production. Il est nécessaire d'utiliser un hébergeur et un espace serveur.

Dans le cadre de ce projet, j'ai décidé d'utiliser la plateforme de cloud computing Heroku, permettant d'externaliser l'hébergement de l'application.

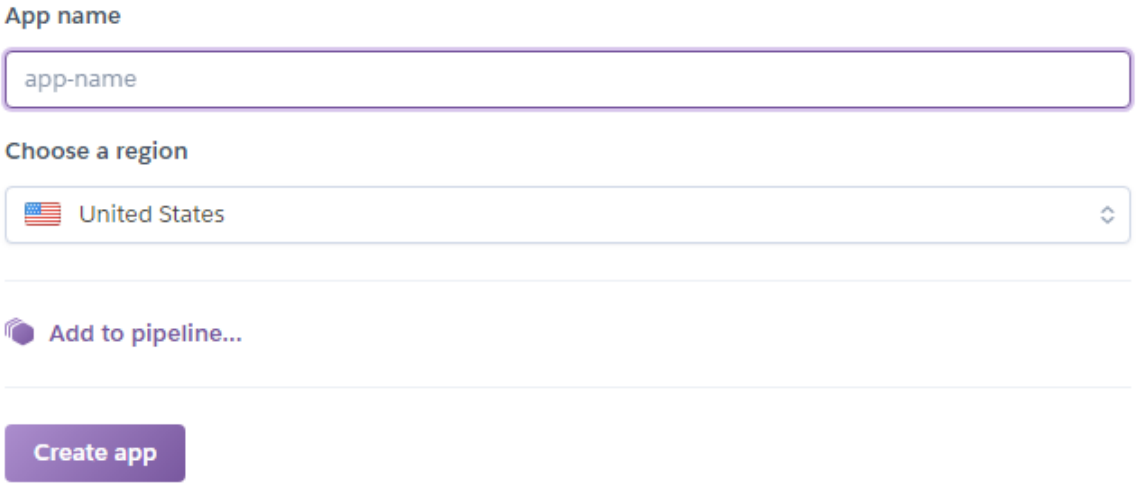
Il est également possible via cette plateforme de gérer la base de données, l'hébergement, et bien plus via les outils proposés.

Liste des étapes pour le passage en production du projet :

- Création d'une nouvelle application Heroku.
- Mise en relation du dépôt GitHub avec l'application.
- Mise en place d'une intégration continue via un déploiement automatique de la branche principale.
- Définition des commandes à exécuter à chaque déploiement au sein du projet symfony.
- Migration de la base de données via un add-on fourni par Heroku.
- Définition des variables d'environnement de configuration d'accès à la nouvelle base de données.
- Mise en place du script de migration de la base de données.
- Déploiement sur Heroku.

CRÉÉR UNE APPLICATION HEROKU

Création d'une nouvelle application sur Heroku :

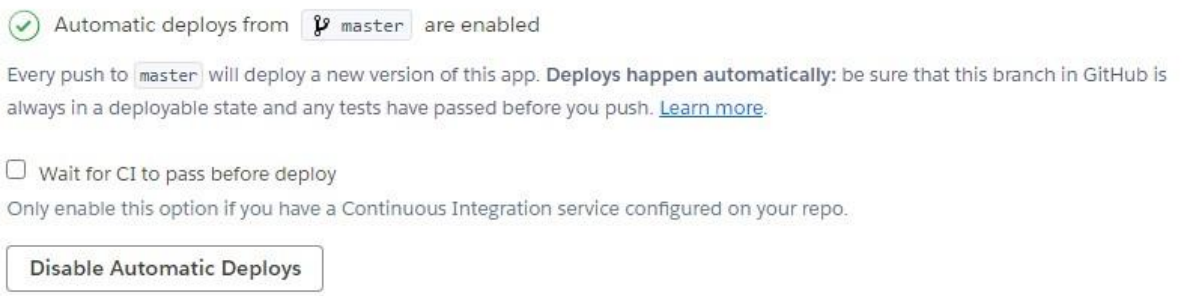


COMMUNICATION GITHUB/HEROKU

Connexion du dépôt Git avec l'application créée :

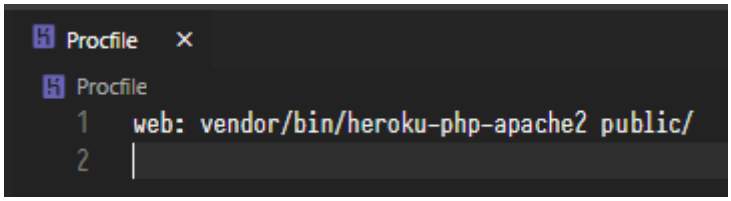
AUTOMATISATION DU DÉPLOIEMENT

Déploiement automatique configuré sur Heroku :



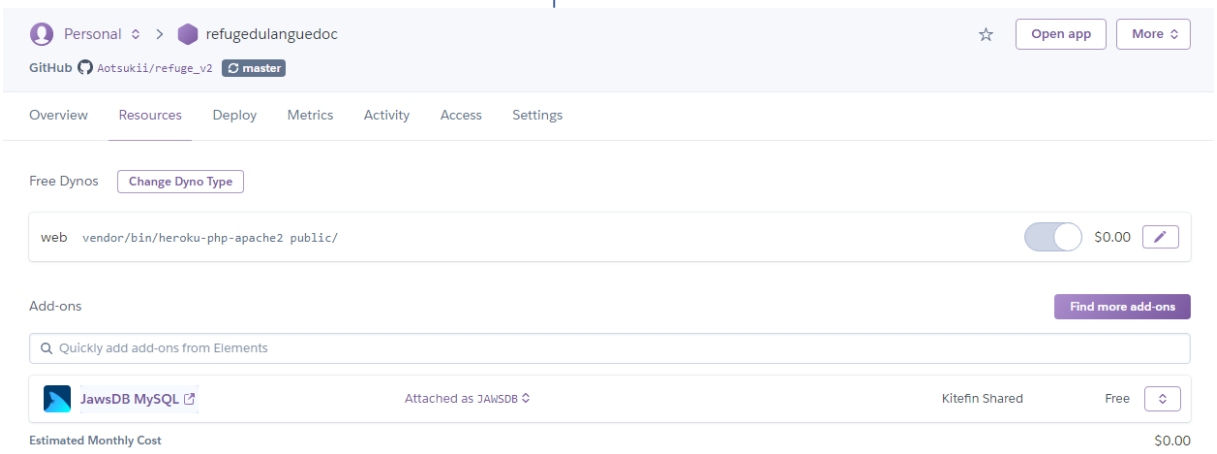
COMMANDES EXECUTÉES

Spécification du démarrage d'un serveur web apache 2 dans le fichier « procfile » :



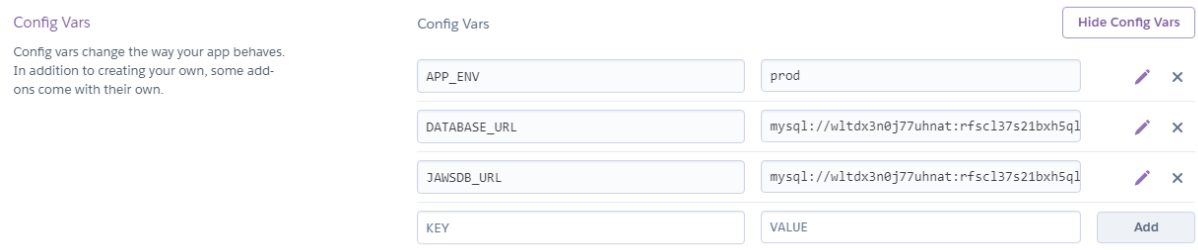
MISE EN PLACE DU SGBD

Utilisation du SGBD JawsDB fourni par Heroku :



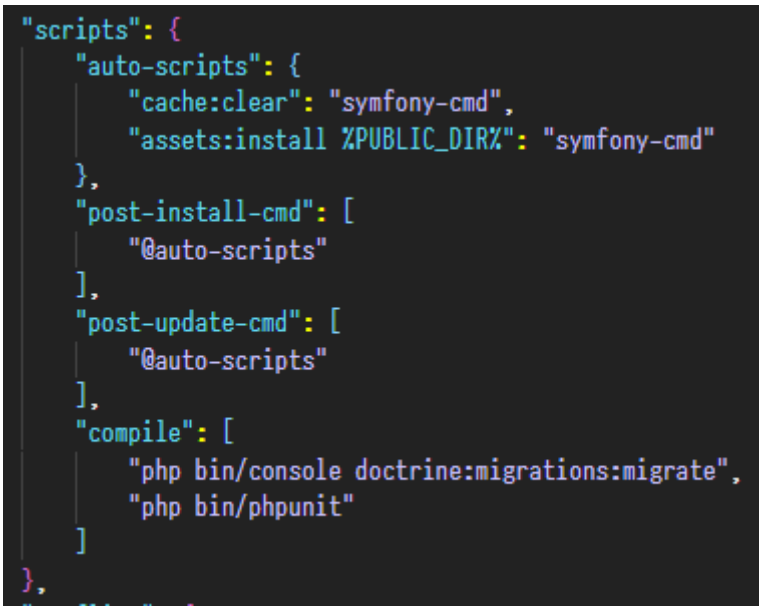
CONFIGURATION DES VARIABLES D'ENVIRONNEMENT

Gestion des variables utilisées pour la connexion à la BDD :



SCRIPT DE MIGRATION

Mise en place des commandes à effectuer lors des déploiements au sein du fichier « composer.json » :

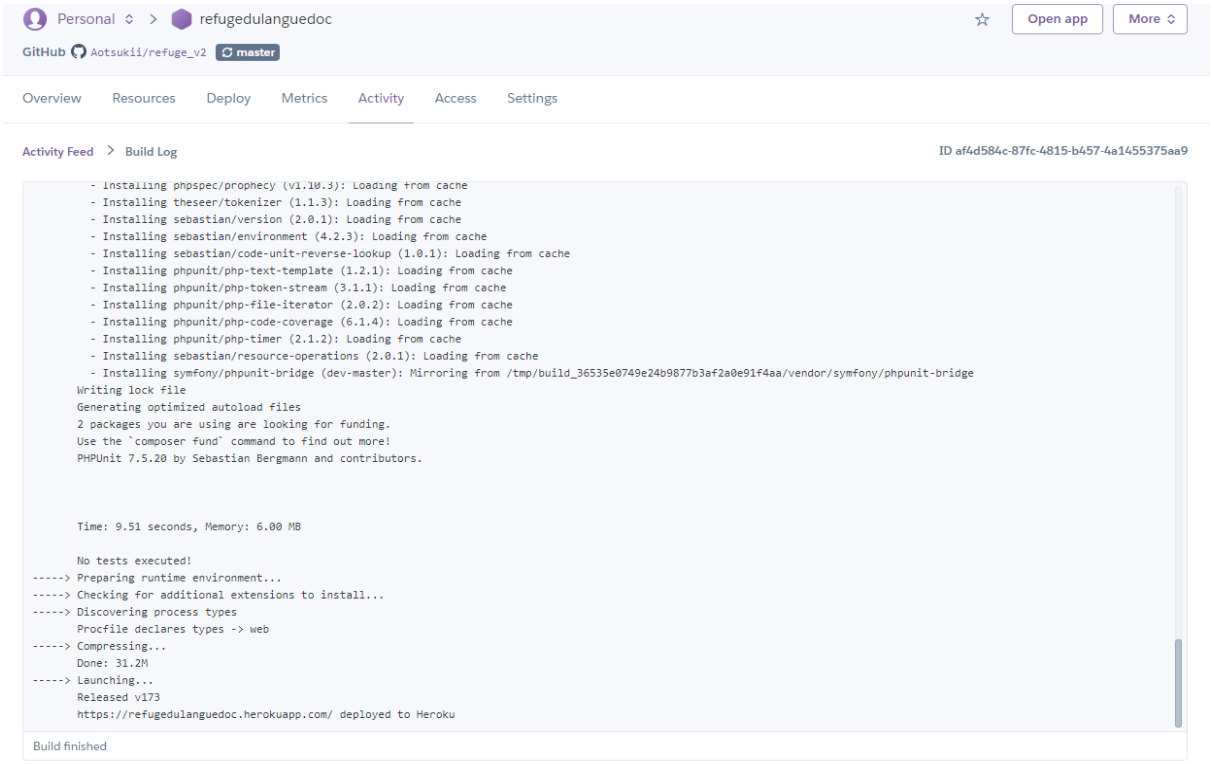


Les commandes situées dans le tableau « compile » seront appelées à chaque déploiement, exécutant respectivement une migration de la base de données si nécessaire ainsi que l'exécution des jeux de tests unitaires/fonctionnels.

Les tests sont bloquants lors du déploiement : un test échoué empêchera le déploiement, dans une optique d'intégration continue.

DÉPLOIEMENT SUR HEROKU

Logs de déploiement de l'application sur Heroku :



SOURCES

SOURCES

Liste des ressources m'ayant aidé tout au long du projet. Le développement de cette application m'a demandé de me documenter sur différents aspects,

CONCEPTION

- Recherche d'outils : maquettage, création de base de données, création de diagrammes,
- Framework pour le développement…

DÉVELOPPEMENT

- Documentation officielle du framework Symfony
  - https://symfony.com/doc/current/index.html
- Documentation officielle du framework PhP
  - https://www.php.net/manual/fr/index.php
- Documentation officielle de Twig
  - https://twig.symfony.com/doc/3.x/
- StackOverflow pour de nombreux soucis rencontrés
  - https://stackoverflow.com/

SÉCURITÉ

- Recherches sur les meilleures pratiques à aborder
- Récupération d'informations dans mes cours RGPD
- Consultation du site de la CNIL pour plus d'informations
  - https://www.cnil.fr/professionnel
- Documentation Symfony concernant son module sécurité
  - https://symfony.com/doc/current/security.html

TESTS

- Documentation officielle de PHPUnit
  - https://phpunit.readthedocs.io/fr/latest/
- Documentation Symfony concernant son module de tests
  - https://symfony.com/doc/current/testing.html
- Cours openClassrooms sur PHPUnit
  - https://openclassrooms.com/fr/courses/4087056-testez-et-suivez-letat-de-votre-application-php/4419446-premiers-pas-avec-phpunit-et-les-tests-unitaires

PRODUCTION

- Documentation Heroku
  - https://devcenter.heroku.com/

CONCLUSION

CONCLUSION

DIFFICULTÉS RENCONTRÉES LORS DU PROJET

Ce projet a été pour moi un défi complexe, mais stimulant. N'effectuant pas dans mon entreprise de développement web, il m'a fallu trouver une idée en partant de zéro, définir tous les besoins de façon réaliste, et m'initier au développement web grâce au framework Symfony, que j'ai utilisé pour la première fois sur un projet de cette échelle.

D'autre part, tout le travail sur ce projet a été réalisé en parallèle de mes cours et de mon emploi, demandant de fait une rigueur et une régularité dans mon rythme de travail.

POINTS POSITIFS DU PROJET

Le sujet du projet, la gestion d'un refuge animalier, me tient tout particulièrement à cœur. Il a donc été plus facile pour moi de m'immerger dans des longues séances de travail les soirs, une fois rentré de mon entreprise ou après les cours.

Définir la totalité de ce projet en partant de zéro m'a permis de bien comprendre la nécessité d'une méthodologie rigoureuse, tant pendant la période de conception que tout au long de la phase de développement.

Je finis ce projet avec une confiance renouvelée en mes compétences de concepteur et développeur d'applications.

ÉVOLUTIONS FUTURES

Ayant pour projet de proposer cette application à l'association de refuge animalier dans laquelle je suis déjà engagé en tant que famille d'accueil, il me faudra optimiser a rapidité de l'application. Il me faudra également implanter réellement la gestion des paiements/donations, peut-être via un module paypal, ainsi qu'un suivi des adoptions plus poussé, avec prise de rendez-vous d'adoption.