

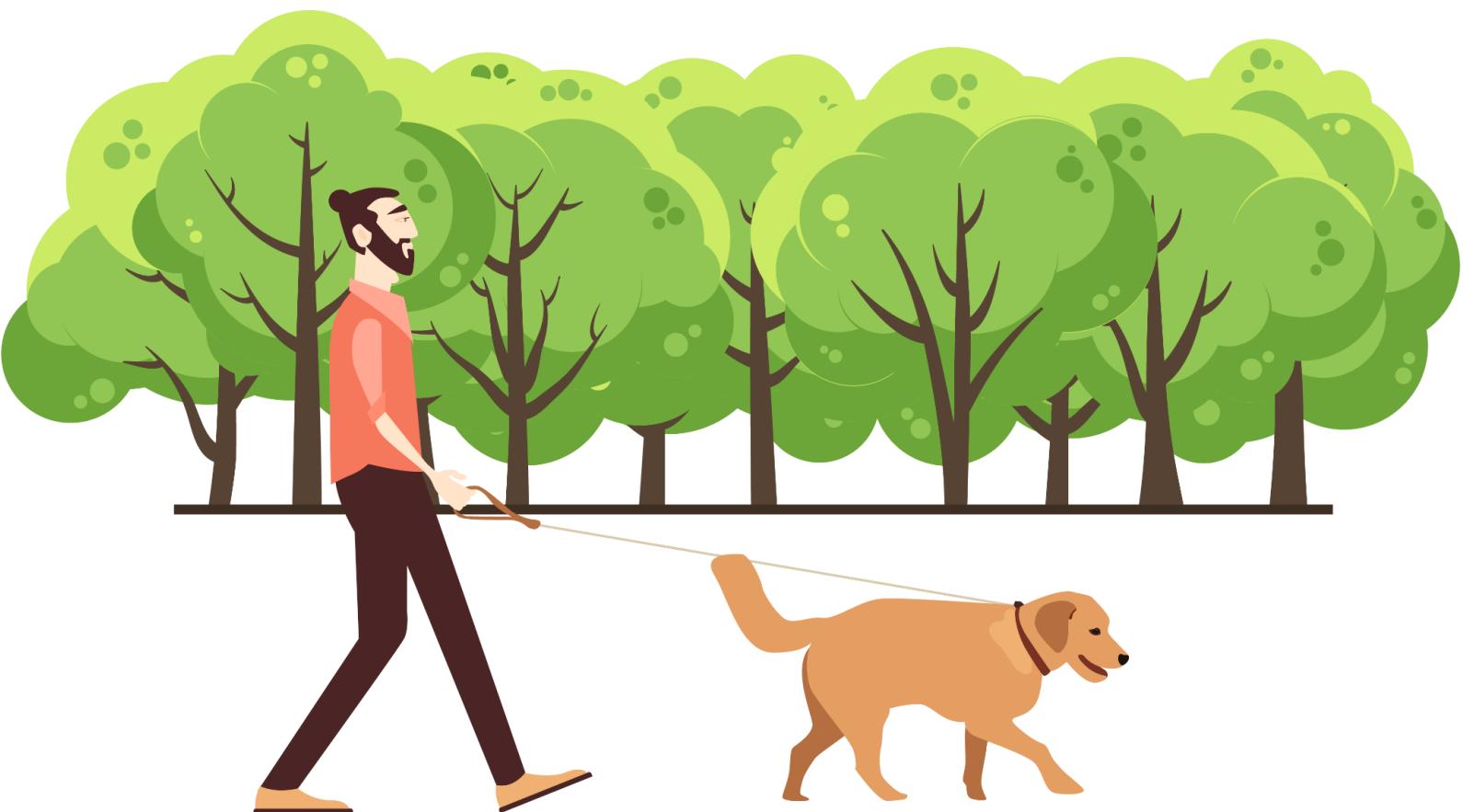
# TOUT'O'POILS

Application d'aide à la gestion du bénévolat en  
refuge animalier

*Projet réalisé dans le cadre de la présentation au*  
**Titre Professionnel Développeur Web et Web Mobile**

*présenté par*

**Bernard ARROUES**  
O'clock - Promotion Kelvin





# SOMMAIRE

<b>SOMMAIRE</b>	<b>3</b>
<b>INTRODUCTION</b>	<b>5</b>
<b>LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET</b>	<b>6</b>
I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	6
A. <i>Maquetter une application</i>	6
B. <i>Réaliser une interface utilisateur web statique et adaptable</i>	6
C. <i>Développer une interface utilisateur web dynamique</i>	6
II. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	6
A. <i>Créer une base de données</i>	6
B. <i>Développer les composants d'accès aux données</i>	7
C. <i>Développer la partie back-end d'une application web ou web mobile</i>	7
<b>RÉSUMÉ DU PROJET</b>	<b>8</b>
<b>CAHIER DES CHARGES</b>	<b>8</b>
I. Besoins et objectifs de l'application	9
A. <i>Besoins</i>	9
B. <i>Objectifs</i>	10
C. <i>Cibles</i>	11
II. Users Stories	12
III. Arborescence	13
IV. MVP	14
A. <i>Bénévole</i>	14
B. <i>Administrateur</i>	14
V. fonctionnalités détaillées des pages	15
VI. Evolutions potentielles	18
A. <i>Bénévole</i>	18
B. <i>Administrateur</i>	19
VII. Wireframes	20
VIII. Charte graphique et logo	26
IX. Exemples de maquettes	27
<b>SPÉCIFICATIONS TECHNIQUES</b>	<b>29</b>
I. Technologies	29
II. Navigateurs compatibles	29
III. Possibilités de déploiement	30
IV. Création de la base de données	33
A. <i>MCD</i>	33
B. <i>MLD</i>	33
C. <i>Dictionnaire des données</i>	34
V. Routes front et back	36
A. <i>Back-end</i>	36

<i>B. Front-end</i>	37
<b>RÉALISATIONS PERSONNELLES</b>	<b>39</b>
I. Upload d'image	39
<i>A. Back</i>	39
<i>B. Front</i>	43
II. Bouton de départ en balade	44
<b>PRÉSENTATION DU JEU D'ESSAI</b>	<b>48</b>
<b>VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE</b>	<b>53</b>
I. Veille technologique	53
<i>A. Prisma</i>	53
<i>B. SDK S3 et Multer</i>	53
<i>C. Nodemailer et serveur SMTP</i>	54
II. Veille de sécurité	54
<i>A. JWT et cryptage du mot de passe utilisateur</i>	54
<i>B. Utilisation des captchas et sécurisation des requêtes</i>	55
III. Processus de recherche et traduction	55
<i>A. Processus</i>	55
<i>B. Ressource en anglais</i>	57
<i>C. Traduction effectuée</i>	57
<b>CONCLUSION</b>	<b>58</b>
<b>ANNEXE</b>	<b>59</b>

# INTRODUCTION

Depuis tout jeune, je suis passionné par les technologies et en particulier la programmation. Bien qu'ayant suivi un parcours m'éloignant de l'univers de l'informatique, j'ai continué pendant longtemps à m'auto former sur mon temps libre sur les sujets liés à la programmation. Grâce à cet apprentissage individuel, j'ai pu continuer la pratique du code et à suivre régulièrement les actualités du secteur.

Après une certaine lassitude de mon précédent domaine d'activité, j'ai choisi de trouver une voie qui me permettrait de valider l'acquisition d'une certaine expérience. Après quelques recherches sur les moyens d'y parvenir, j'ai décidé de m'engager auprès de l'école O'clock. Au lieu de choisir de suivre la formation complète, j'ai effectué un test de positionnement pour accéder directement à la seconde partie de la formation, avec une spécialisation back-end.

Après avoir suivi ce parcours et pour conclure la formation, l'école O'clock met en place l'Apothéose. L'Apothéose consiste à réaliser un projet, pendant un mois et en groupe avec d'autres apprenants de notre promotion. C'est donc à cette occasion que nous avons réalisé le projet Tout'o'poils avec mes quatres collègues.

Au-delà de l'acquisition de nouvelles compétences et la validation d'une expérience, je souhaitais également apprendre à travailler en groupe sur un projet. Ayant toujours réalisé des petits projets de façon autonome, je souhaitais acquérir des bonnes pratiques pour la réalisation d'un travail de groupe, et comprendre l'organisation et le fonctionnement d'une équipe pour un projet.

En somme, cette période de réalisation du projet Tout'o'poils m'a permis d'acquérir ces nouvelles compétences, tout en travaillant avec une équipe de quatres personnes sur un sujet qui, initialement, ne m'aurait pas forcément intéressé si j'avais eu à le faire de façon autonome.

# LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

## I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

### A. *Maquetter une application*

En amont de la réalisation à proprement parler de notre application, nous avons choisi dans un premier temps de travailler sur la réalisation des documents de conception.

Dans ce but, nous avons donc réalisé en premier lieu des users stories pour définir très clairement ce que pourront faire les utilisateurs sur notre application. Puis nous avons réalisé les wireframes du projet, aux formats mobile et desktop pour imaginer la structuration de nos pages, puis enfin nous avons réalisé une charte graphique et des maquettes pour voir concrètement à quoi allait ressembler notre application.

### B. *Réaliser une interface utilisateur web statique et adaptable*

Ce projet étant conçu comme un outil quotidien à la réalisation des actions de bénévolat au sein d'une antenne locale de la SPA. Au vu de la diversité des bénévoles de la SPA, notamment au niveau de l'âge, il était nécessaire d'avoir une interface simple et facile d'utilisation, peu importe le niveau de compétence en informatique.

De plus, cette application étant utile "sur le terrain" pour les bénévoles, il était impératif d'avoir une application compatible avec les formats d'écran mobile.

### C. *Développer une interface utilisateur web dynamique*

Au vu des users stories réalisées en amont de la réalisation du projet, il était impératif que notre application dispose d'une interface dynamique, dès lors nous nous sommes très rapidement tournés vers la librairie React.

## II. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

### A. *Créer une base de données*

Tout comme la réalisation du maquettage de l'application, nous avons voulu dès le début organiser la façon dont allaient être structurées nos données. Nous avons donc réalisé commencé par réfléchir aux données que nous souhaitions intégrer à l'application, puis nous avons créé un MCD, un MLD, ainsi qu'un dictionnaire des données. Pour gérer la base de données de notre application, nous avons choisi d'utiliser PostgreSQL.

## *B. Développer les composants d'accès aux données*

Pour réaliser la couche d'accès aux données de notre application, nous avons utilisé Prisma. Pour assurer un versionning de notre base de données, nous nous sommes servis de l'outil de versionning intégré de Prisma.

## *C. Développer la partie back-end d'une application web ou web mobile*

Pour réaliser le back-end de notre application, nous avons utilisé express. Pour réaliser celà, nous avons organisé notre application avec différentes routes, liées à des controllers. Nous avons également créé plusieurs middleware, notamment pour l'authentification ou la validation des données.

## RÉSUMÉ DU PROJET

Chaque année des milliers de chiens et de chats sont abandonnés et se retrouvent dans un refuge associatif. Ces centres sont gérés au quotidien par des équipes présentes tout au long de l'année mais aussi par des bénévoles. L'organisation interne en devient complexe.

Notre application web nommée « ToutOPoils » a été créée dans le but d'aider ces associations qui gèrent des animaux abandonnés, à l'instar des SPA (Sociétés Protectrice des Animaux), des fondations (fondation Brigitte Bardot)..., à organiser au mieux la gestion de ses animaux, de ses équipements et de ses bénévoles.

Les animaux étant nombreux, les associations n'ont pas forcément de moyens humains suffisants pour effectuer les tâches quotidiennes récurrentes telles que la promenade, les jeux, le nettoyage.... Elles comptent alors énormément sur l'aide des bénévoles dont elles n'ont pas non plus les moyens de les accompagner dans la prise en charge des missions. Ces derniers, après avoir pris connaissance des missions attendues, doivent être autonomes.

Ainsi, cette application a été pensée pour permettre à l'ensemble des bénévoles présents de se positionner en toute autonomie sur les tâches à effectuer et d'en gérer son état. En d'autres termes, ils pourront veiller à la traçabilité des tâches. Fini les doublons, fini les laisser pour compte ou les préférences. Tout ce petit monde sera logé à la même enseigne. Cette application se veut avant tout pratique, par un gain de temps sur l'accès à l'information, mais aussi un bon moyen de communication virtuelle car les bénévoles pourront informer instantanément et à distance chaque action effectuée.

En définitive, cette solution favorise la cohérence du travail rendu, un gain de temps organisationnel et une traçabilité. Mais pour l'heure, ce projet est à destination d'une SPA précise.

# CAHIER DES CHARGES

## I. Besoins et objectifs de l'application

### A. Besoins

Pour mieux comprendre le besoin de la SPA, plusieurs recherches sont nécessaires pour connaître l'utilité du projet. Il faut commencer par connaître l'organisation interne de cette dernière.

En allant sur le site de la SPA, nous pouvons trouver toutes les missions bénévoles proposées.

Les missions bénévoles à la SPA diffèrent d'un site à l'autre. Chaque SPA n'a pas forcément les mêmes besoins, mais voici toutes les missions à destination des bénévoles :

- **Bien-être des animaux sur site**
  - ❖ Accompagnement du chien
  - ❖ Accompagnement du chat
- **Entretien et confort des animaux**
  - ❖ Entretien et nettoyage des chemils
  - ❖ Entretien et nettoyage des chatteries
  - ❖ Entretien du site, petits travaux et bricolage
- **Support bureautique**
  - ❖ Accueil et information du public
  - ❖ Appui administratif
  - ❖ Assistance communication digitale
- **Animation et sensibilisation à la cause animale**
  - ❖ Animateur pédagogique jeunesse
  - ❖ Encadrant club jeunes
  - ❖ Animation sociale
  - ❖ Animations, évènements et collectes
- **Bien-être des animaux hors site**
  - ❖ Chats libres
  - ❖ Accompagnement transferts d'animaux
  - ❖ Famille relais
  - ❖ Visiteur post-adoption
  - ❖ Visiteur pré-adoptions équidés et animaux de la ferme
- **Lutte contre la maltraitance**
  - ❖ Référent délégué-enquêteur
  - ❖ Délégué-enquêteur

## B. Objectifs

Nous comprenons que pour optimiser les ressources humaines, les informations à prendre connaissance sont :

- Le bénévole doit pouvoir prendre connaissance des missions attendues par l'association.
- Les missions sont visibles via un support de communication mis en place par l'association.
- Le bénévole devra être autonome sans avoir à solliciter les employés permanents déjà occupés.
- Les bénévoles organisent la répartition des tâches entre eux.
- Le bénévole est en mesure de connaître la tâche déjà effectuée par un collègue.
- L'association a déterminé le principe d'organisation ce qui permet aux bénévoles de s'organiser entre eux.

Après plusieurs recherches, aucun logiciels ou applications n'existent sur ce type de projet, il existe bien-sûr la gestion d'une association mais qui ne gère pas le planning des bénévoles.

Nous pouvons éventuellement prendre simplement un planificateur. Dans ce cas, le logiciel « QO-EZION » est approprié car il gère le planning des bénévoles mais uniquement sur de l'événementiel. L'inconvénient pour la SPA avec ce type de logiciels serait éventuellement la création d'événement pour permettre la planification et de charger une personne pour organiser le planning de chaque bénévole. Dans la foulée, du même genre que « QO-EZION », il y a « RECREWTEER », « VOLUNTEO »...

Revenons sur le site de la SPA qui regroupe tous les centres du territoire français et où nous pouvons trouver toutes les informations utiles au développement de notre produit.

En effet, nous pouvons trouver les centres qui recherchent par exemple un bénévole pour l'accompagnement des chiens et des chats. La recherche renvoie au centre SPA de BRUGHEAS (03) et au centre SPA de CROZON (29). Respectivement, il y a 97 animaux à adopter dont 36 sont des chiens et 60 sont des chats et 211 animaux dont 85 sont des chiens et 117 sont des chats.

Par ces chiffres, nous pouvons déjà prendre connaissance de la taille de la structure et le besoin en ressources humaines et par conséquent le besoin en bénévoles. Mais aussi que la SPA a une forte occupation de chiens et de chats, les autres espèces d'animaux représentent environ moins de 5% des effectifs animaliers.

Ainsi, l'objectif est d'optimiser l'organisation interne avec peu de ressources humaines grâce à l'application.

## C. Cibles

Afin d'avoir une bonne représentation de la cible de nos utilisateurs, nous avons établi un persona.

### **Sébastien Lauret** 31 ans, Toulouse, Commercial



**Equipement informatique :** tablette ipad, iphone 14  
**Ce qu'il recherche :** Apporter son expérience du commerce pour aider la spa sur des actions  
**Sa navigation :** Il va aller sur la page animation  
**Ce qu'il attend :** Que les informations soient claires concernant les événements  
**Prérequis :** Le site doit avoir une bonne expérience utilisateur

### **Eliane Petit** 75 ans, Paris, Retraité de la fonction publique



**Equipement informatique :** Desktop windows 10, mobile android  
**Ce qu'elle recherche :** S'occuper des chats et leur donner de l'amour  
**Sa navigation :** Elle va aller sur la page d'accès aux chats  
**Ce qu'elle attend :** Une application facile d'utilisation  
**Prérequis :** Le site doit avoir une navigation simple et cohérente entre la version mobile et desktop

### **Sarah Aïtbaziz** 28 ans, Paris, infirmière



**Equipement informatique :** tablette et mobile android  
**Ce qu'elle recherche :** Donner de son temps libre pour se balader avec les chiens  
**Sa navigation :** Va aller sur la liste de chien et faire une balade  
**Ce qu'elle attend :** Une application intuitive et esthétique  
**Prérequis :** Le site doit avoir un accès rapide aux balades

### **Willem Feval** 45 ans, Lyon, référent SPA



**Equipement informatique :** desktop window 10, iphone 11  
**Ce qu'il recherche :** Être efficace dans sa mission envers les animaux  
**Sa navigation :** Accès administrateur  
**Ce qu'il attend :** Gagner du temps et accéder rapidement aux informations  
**Prérequis :** Le site doit être réactif et pratique dans la navigation

## II. Users Stories

### Légende :

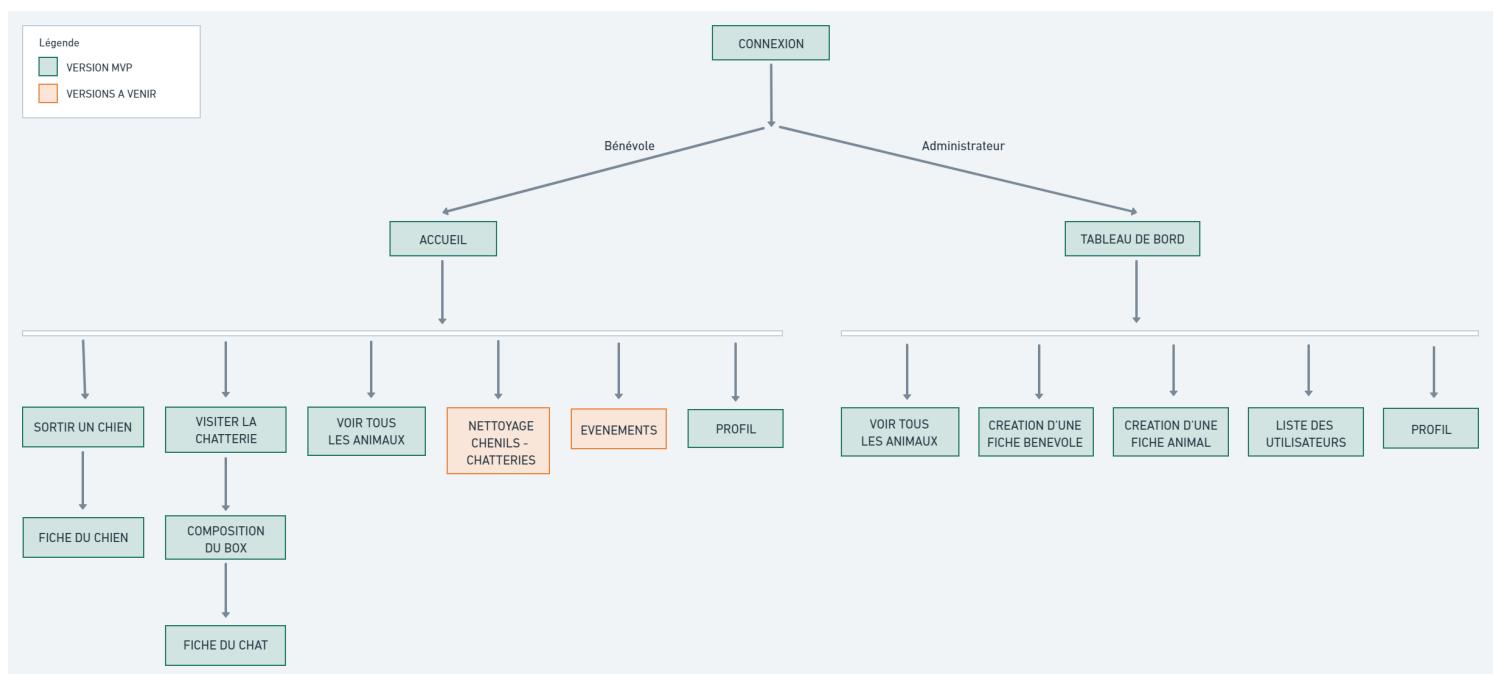
MVP

Versions à venir

En Tant que	Je souhaite	Afin de
visiteur	me connecter à l'appli	accéder aux différents services
visiteur	me déconnecter de l'appli	pour laisser la place à d'autres utilisateurs
utilisateur connecté	accéder à mon profil	modifier mes informations personnelles
bénévole	voir la liste des chiens	voir ceux qui doivent être sortis
bénévole	connaître les tâches prioritaires	d'apporter mon aide rapidement
bénévole	pouvoir filtrer les caractéristiques de l'animal	choisir l'animal correspondant à mes préférences
bénévole	savoir ou je peux voir les chats	leur rendre visite et jouer avec
bénévole	consulter la liste des animaux	pour voir leur fiche
bénévole	savoir si un animal précis a été adopté	me renseigner sur mon chouchou (favoris)
bénévole	signaler un problème	informer au plus vite les référents spa
bénévole	savoir si on a besoin de moi pour une collecte ou autre	d'apporter mon aide
administrateur	suivre l'état de sortie des chiens	savoir si tous les chiens sont sortis
administrateur	ajouter un animal dans le refuge	pour qu'il puisse avoir un suivi
administrateur	attribuer une cage ou un box à un animal	le localiser facilement
administrateur	savoir si les cages ou box sont nettoyés	suivre la propreté des locaux
administrateur	ajouter des événements	informer les bénévoles
administrateur	voir s'il y a des problèmes avec des animaux	agir au plus vite pour sa santé
administrateur	ajouter un bénévole	qu'il puisse utiliser l'application
administrateur	consulter la liste des bénévoles	pour voir les utilisateurs inscrits

administrateur	libérer des cages lorsqu'un animal est adopté	ajouter un autre animal a la place
----------------	---	------------------------------------

### III. Arborescence



## IV. MVP

Notre Produit Minimum Viable s'est naturellement construit autour de la sortie du chien et de la visite des chats qui sont le cœur des activités des bénévoles des refuges. Pour cela une authentification sécurisée est primordiale, ainsi que les fonctionnalités liées à l'administrateur qui permettent que les informations données au bénévole soient disponibles et mises à jour.

### **Voici donc les fonctionnalités présentes dans notre MVP :**

- l'utilisateur (bénévole ou administrateur) doit pouvoir s'authentifier avec un email et un mot de passe
- l'utilisateur doit pouvoir consulter les mentions légales et la politique de confidentialité du site

#### *A. Bénévole*

- arrive sur la page accueil : il doit pouvoir sortir un chien, visiter la chatterie, accéder au menu ou accéder à son profil
- pour sortir un chien, il doit pouvoir voir la liste des chiens et pouvoir filtrer cette liste selon certains critères
- une fois sur cette liste affichée, il doit pouvoir cliquer sur le chien de son choix et ainsi voir sa fiche complète
- sur la fiche du chien, il doit pouvoir consulter les informations (caractéristiques du chien, historique des balades) et valider la sortie par un bouton
- une fois le bouton "partir en balade" cliqué, une pop-up de confirmation apparaît
- à la fin de la balade, il doit pouvoir cliquer sur le bouton "terminer la balade", et pouvoir entrer un commentaire (pop-up) sur son expérience avec le chien
- pour visiter un chat, il doit pouvoir voir la liste des boxes et cliquer sur le box de son choix pour voir les chats présents, valider son intention de visite et laisser un commentaire sur sa visite (pop-up)
- en cliquant sur la fiche du chat, il doit pouvoir consulter les informations (caractéristiques du chat, historique des visites)

Dans le menu:

- il doit pouvoir se rendre dans la rubrique "voir les animaux" et filtrer selon l'espèce pour consulter la fiche d'un animal
- il doit pouvoir accéder à son compte et modifier ses informations dans la rubrique "profil" dont le mot de passe

#### *B. Administrateur*

- arrive sur la page tableau de bord, il doit pouvoir :

- consulter la liste des animaux (comme pour le bénévole)
- créer une nouvelle fiche bénévole
- créer une nouvelle fiche animal pour le rentrer dans le refuge
- consulter la liste des utilisateurs
- accéder à son profil en cliquant sur l'avatar (comme pour le bénévole)
- pour inscrire un bénévole il doit pouvoir récolter les informations nécessaires et lui envoyer un mail de confirmation avec les identifiants
- pour entrer un animal dans le refuge, il doit pouvoir enregistrer les informations et caractéristiques de l'animal

## V. fonctionnalités détaillées des pages

Les pages/routes sont sécurisées et donc **accessibles seulement à partir du moment où l'utilisateur est connecté**. Seule la page de connexion est accessible sur le net par tout le monde.

**Au préalable, le bénévole doit s'inscrire au refuge** auprès d'un employé (administrateur). Il reçoit par mail un message de bienvenue avec ses identifiants. Il pourra s'il le souhaite modifier son mot de passe dans sa page profil.

### ➤ Page connexion :

Pour le moment l'utilisateur n'a pas l'accès au menu. Il doit compléter les champs :

- email
- mot de passe

Il valide son authentification grâce à un bouton. Il accède à la page d'accueil si c'est un bénévole ou à la page dashboard si c'est un admin.

### ➤ Page accueil :

Cette page est donc propre au bénévole. Dans la partie navigation, il peut maintenant accéder au menu et à sa page profil. Le corps de la page présente deux choix principaux de redirection : Sortir un chien ou Visiter la chatterie.

### ➤ Page sortir un chien :

La page Sortir un chien se présente sous forme de liste. Elle tient compte de l'expérience du bénévole, s'il n'est que "débutant", il sera limité aux chiens sociables et faciles. Les chiens apparaissent **par ordre de priorité de sortie** et sa date de dernière sortie est précisée suivant ces couleurs :

- rouge : urgent, sorti depuis 3 jours
- orange : sorti depuis 2 jours
- vert : sorti la veille

Le chien sorti dans la journée n'apparaît plus dans la liste.

Un filtre permet également au bénévole de choisir le chien selon ses préférences, selon les caractéristiques suivantes :

- sexe
- âge
- gabarit
- tags (correspondant aux tempéraments)

Tout en restant vigilant aux ordres de priorité de sortie des chiens pour n'en délaisser aucun, le bénévole clique sur l'élément correspondant au chien choisi.

Il accède alors à sa fiche.

#### ➤ **Page fiche chien**

Sur la fiche du chien on retrouve :

- son nom
- le numéro de son box
- les informations données par le refuge
- ses caractéristiques sous forme de tags
- sa photo éventuelle
- le bouton "Partir en balade"
- l'historique de ses balades

Après avoir lu les caractéristiques du chien, si le bénévole le souhaite, il peut cliquer sur le bouton "Partir en balade". A ce moment, **le chien est retiré de la liste des sorties** et les autres bénévoles ne peuvent plus le voir. Le bouton "Partir en balade" précédemment cliqué se transforme alors en "Terminer la balade" qui doit être cliqué au retour. Cette action déclenche une pop-up pour entrer un commentaire et son ressenti avec le chien. Si le bouton "Terminer la balade" n'a pas été cliqué, la session sera automatiquement fermée au bout d'une heure.

On affiche dans l'historique les 4 dernières balades. Sur chaque balade, le ressenti est visible avec un tag de couleur :

- bon : vert
- moyen : orange
- mauvais : rouge

Il y a un bouton "voir plus" pour afficher davantage d'historiques. On peut cliquer directement sur une balade pour avoir son détail, celui-ci s'affiche alors en dessous.

#### ➤ **Page visiter la chatterie**

La page Visiter la chatterie présente la liste des boxes de chats. Ici le bénévole peut sélectionner le box numéroté qu'il souhaite visiter en cliquant dessus. Dans ce cas, il est amené sur la page composition d'un box.

#### ➤ **Page composition d'un box**

Cette page présente tous les chats qui composent le box et le bouton permettant de lancer la "Visite du box". Le bénévole peut donc soit cliquer sur une fiche chat pour voir ses caractéristiques ou sur le bouton "Visite du box". Comme pour le fonctionnement des chiens, au clic de ce dernier le bouton devient "Terminer la visite" et **le box est retiré de la page**

**Visiter la chatterie.** A l'action de ce dernier, la pop-up commentaire et ressenti s'affiche. Il sera affiché dans chaque fiche chat composant le box.

#### ➤ **Page fiche du chat**

Sur la fiche du chat on retrouve :

- son nom
- le numéro de son box
- les informations données par le refuge
- ses caractéristiques sous forme de tags
- sa photo éventuelle
- l'historique de ses visites (visite du box auquel il appartient)

Comme pour le fonctionnement des chiens, on affiche dans l'historique les 4 dernières visites. Sur chaque visite, le ressenti est visible avec un tag de couleur :

- bon : vert
- moyen : orange
- mauvais : rouge

Il y a un bouton "voir plus" pour afficher davantage d'historiques. On peut cliquer directement sur une visite pour avoir son détail, celui-ci s'affiche alors en dessous.

#### ➤ **Page voir tous les animaux**

Cette page présente la liste de tous les animaux présents sur le site. Chacun des animaux est contenu dans un élément cliquable (nom et photo éventuelle) qui amène à la fiche de l'animal.

Il est possible de filtrer la liste via l'espèce.

#### ➤ **Page tableau de bord**

Le tableau de bord permet à l'administrateur de choisir ce qu'il souhaite faire. En plus d'avoir accès à son profil dans la partie navigation et à son menu, il peut se joindre aussi directement sur les pages suivantes (éléments cliquables) :

- voir tous les animaux
- création d'une fiche bénévole
- création d'une fiche animal

#### ➤ **Page création d'une fiche bénévole**

Sur la page création d'une fiche bénévole, l'administrateur complète le profil du bénévole avec les champs suivants :

- nom
- prénom
- adresse mail
- mot de passe
- numéro de téléphone
- expérience

Il enregistre ensuite la fiche grâce au bouton "valider".

### ➤ **Page création d'une fiche animal**

Pour créer la fiche d'un animal, l'administrateur doit compléter plusieurs champs:

- son espèce
- son nom
- son numéro de box
- son genre
- son âge
- son poids
- ses caractéristiques sous forme de tags

Il peut aussi préciser des informations utiles pour le bénévole et ajouter une photo.

Il enregistre ensuite la fiche grâce au bouton "valider".

### ➤ **Page liste des utilisateurs**

L'administrateur a accès à la liste des utilisateurs, sous forme de petite carte individuelle. On y retrouve pour chaque utilisateur :

- son nom
- son prénom
- sa photo éventuelle
- son étiquette "bénévole" ou "employé"

### ➤ **Page profil**

Tout utilisateur (aussi bien le bénévole que l'administrateur) peut accéder à ses informations personnelles via la rubrique profil. On y retrouve :

- son nom
- son prénom
- son adresse mail
- mot de passe
- son numéro de téléphone
- sa photo éventuelle
- son expérience **si c'est un bénévole**

Un bouton "éditer" permet de rendre chaque champ modifiable.

## VI. Evolutions potentielles

Il est prévu d'améliorer l'application par des fonctionnalités complémentaires pour une meilleure utilisation et une meilleure expérience utilisateur. Les améliorations seront apportées sur les 2 parties utilisateurs.

### A. Bénévole

- pourra prendre connaissance des autres activités bénévoles telles que tout ce qui attire à l'entretien et le confort des animaux, au support bureautique, à l'animation et la sensibilisation à la cause animale
- pourra peaufiner la liste en ayant la possibilité de mettre en favori son animal préféré et ainsi connaître sa finalité (ex : adoption)

- pourra également signaler l'urgence d'un commentaire déposé sur la fiche de l'animal qui permettra de donner un caractère important et visible rapidement par les référents de la SPA
- pourra savoir en amont si des activités événementielles sont prévues

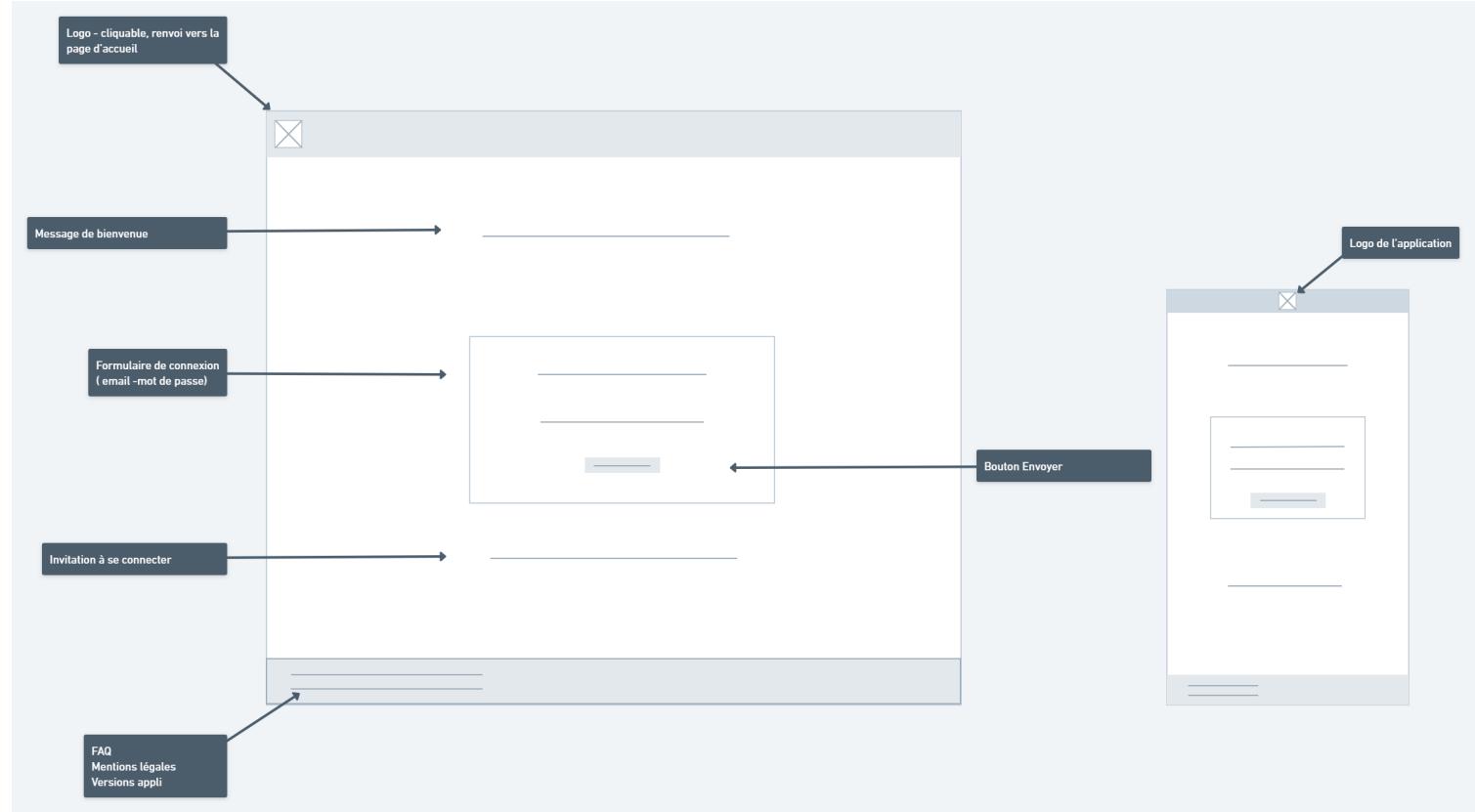
#### *B. Administrateur*

- aura la possibilité de voir si le nettoyage des cages ou des box ont été effectués
- pourra ajouter la race de l'animal dans sa fiche
- pourra créer les futurs événements afin d'informer en amont les bénévoles
- pourra visualiser l'animal qui a été signalé par un bénévole comme étant en danger afin de pouvoir traiter l'information rapidement
- pourra supprimer et/ou archiver un bénévole
- pourra supprimer et/ou archiver un animal
- pourra changer le niveau d'expertise du bénévole

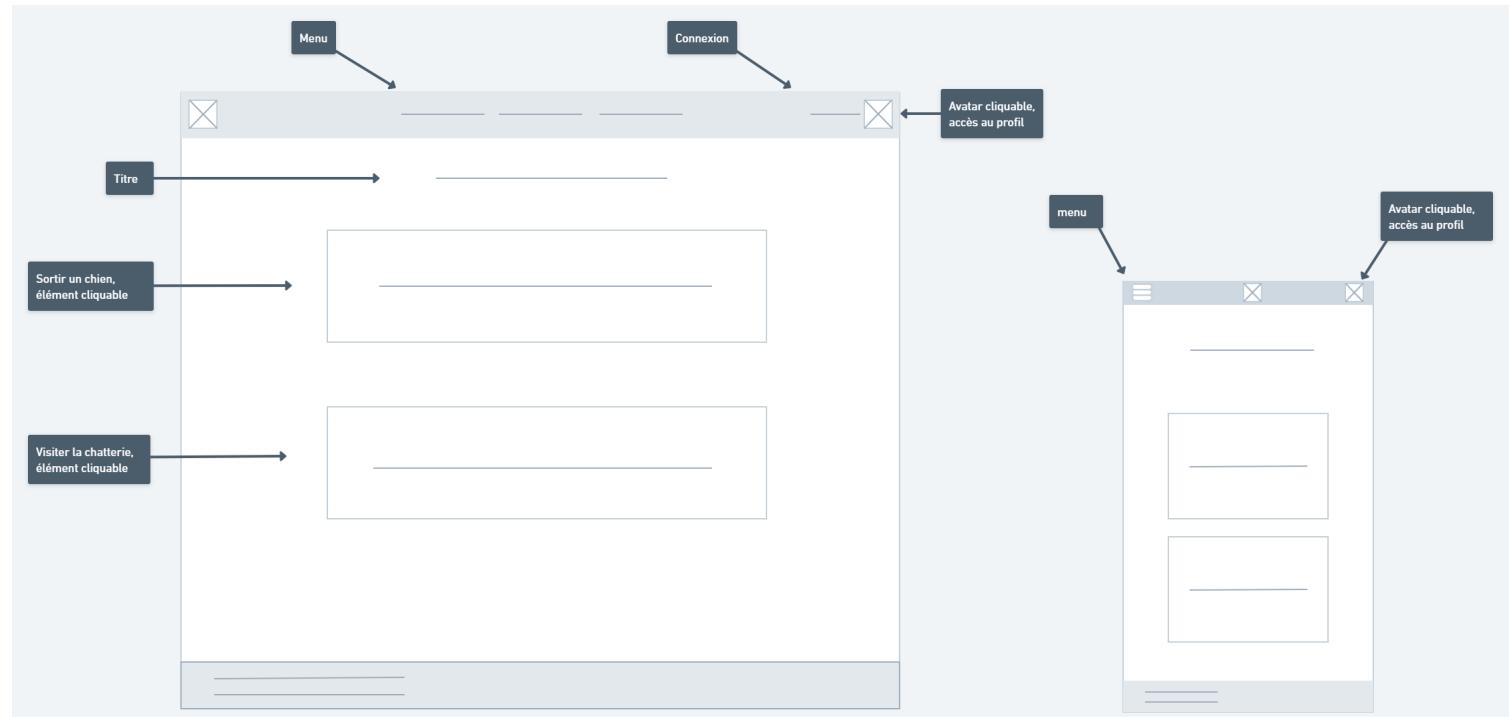
## VII. Wireframes

Pour plus de lisibilité, l'intégralité des wireframes est disponible à l'adresse suivante:  
<https://whimsical.com/top-formats-desktop-mobile-CuVu8V7ifJsVwWaYizz81w>

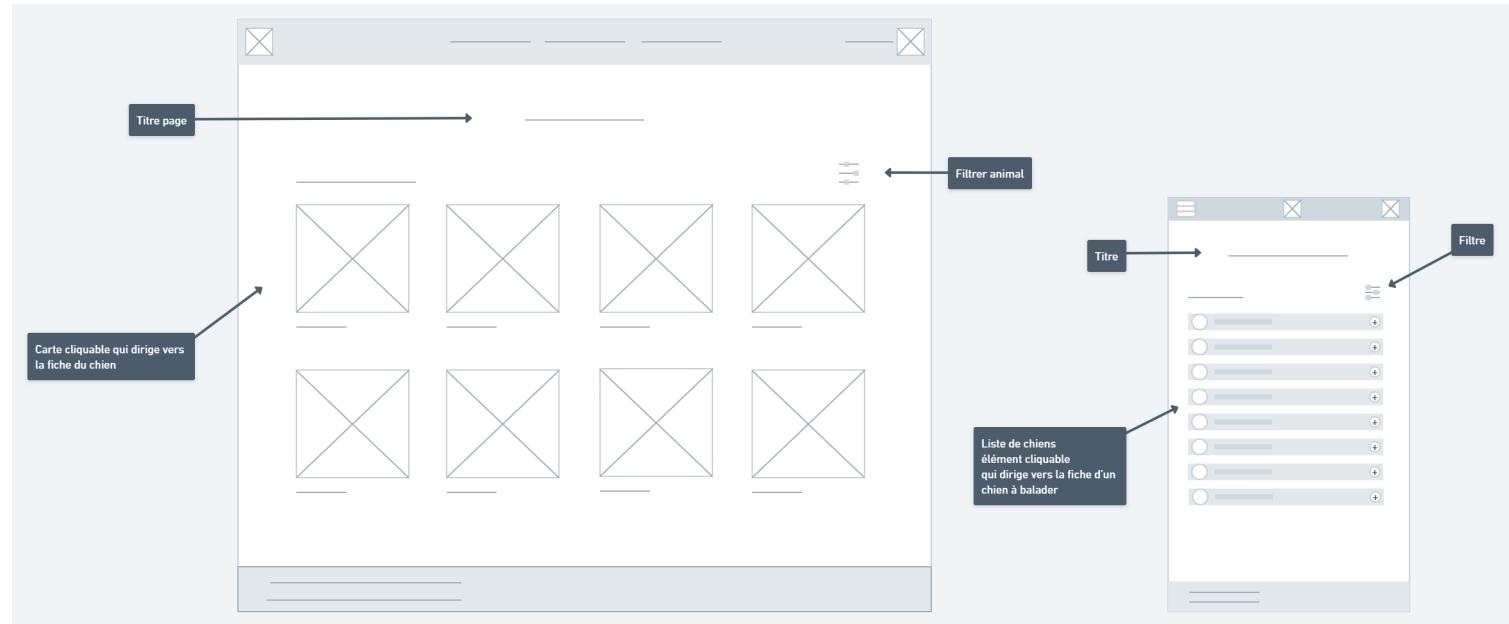
*Écran de connexion*



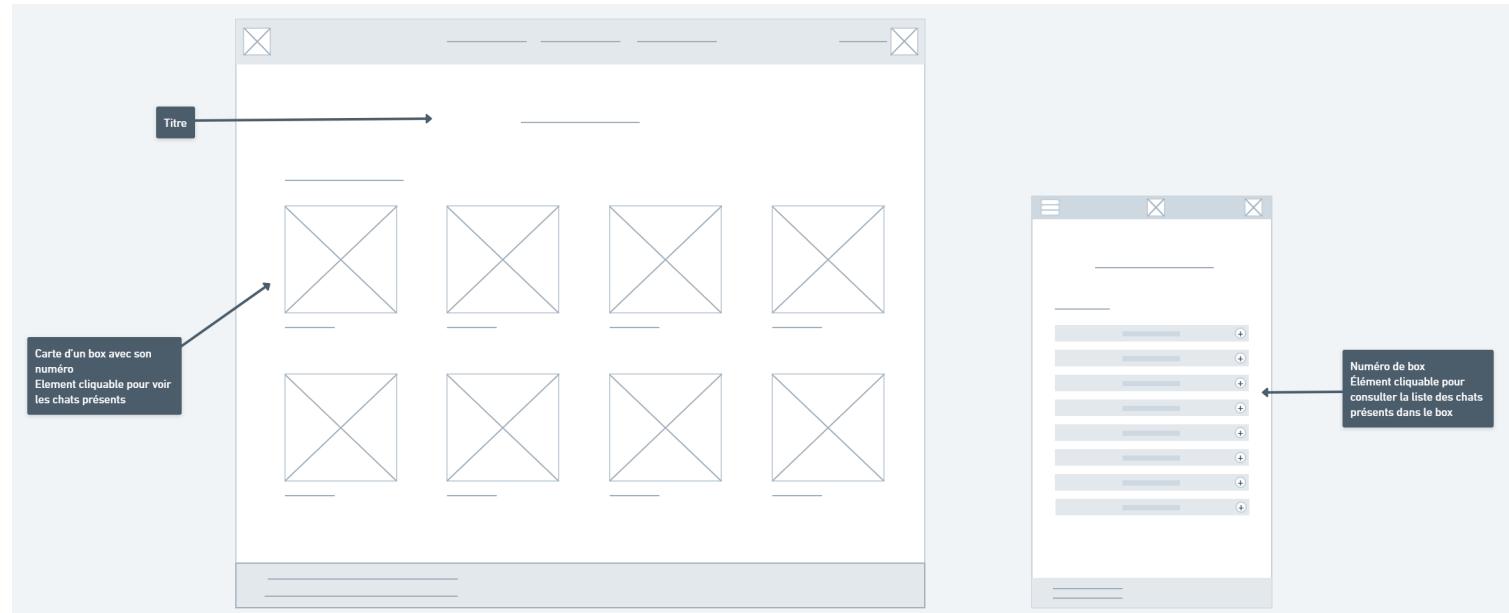
*Accueil*



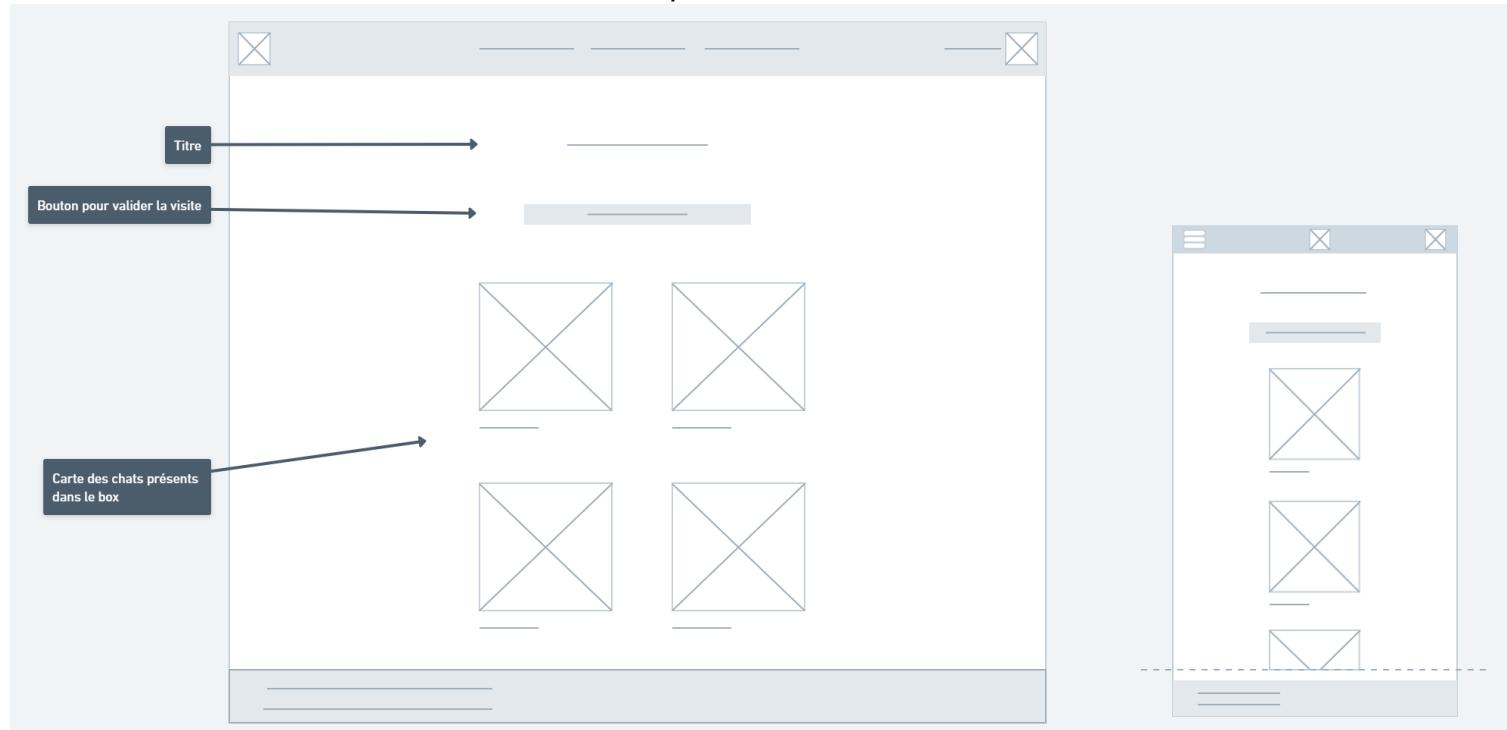
## Sortir un chien



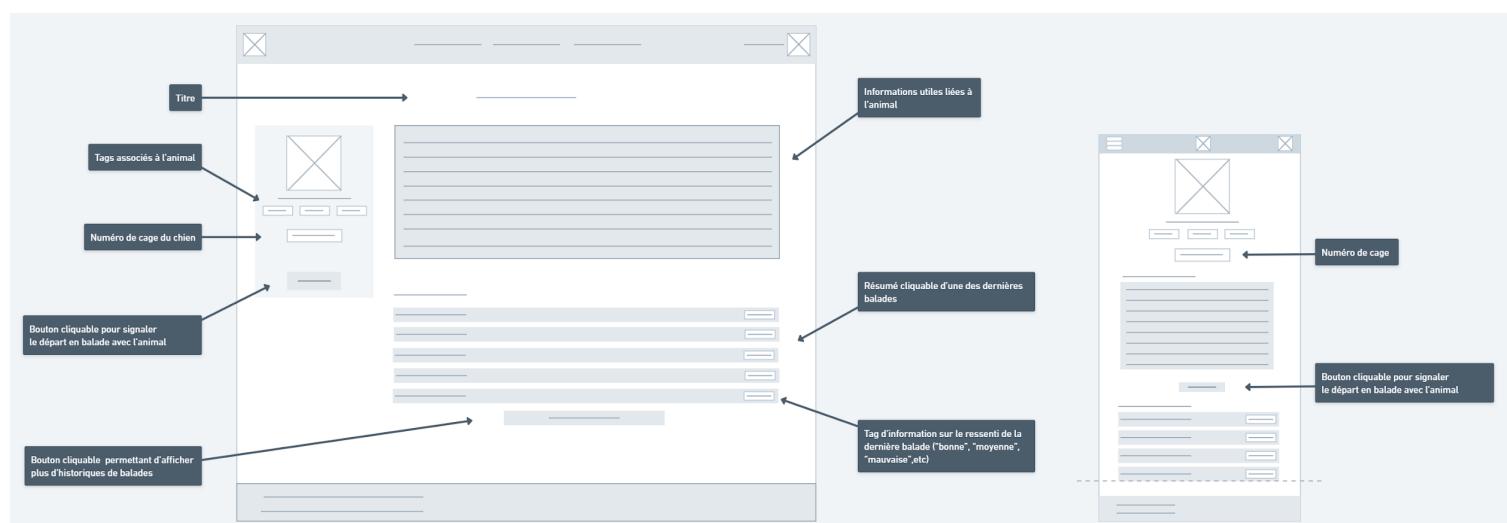
## Liste des box à visiter



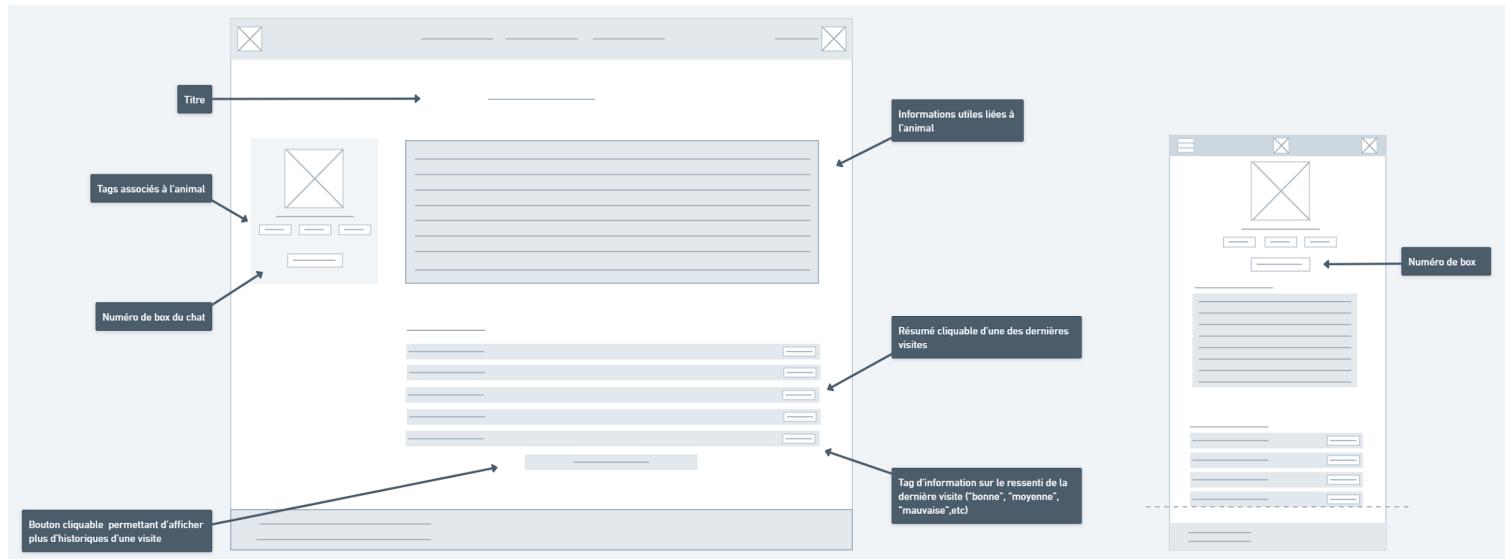
## Composition d'un box



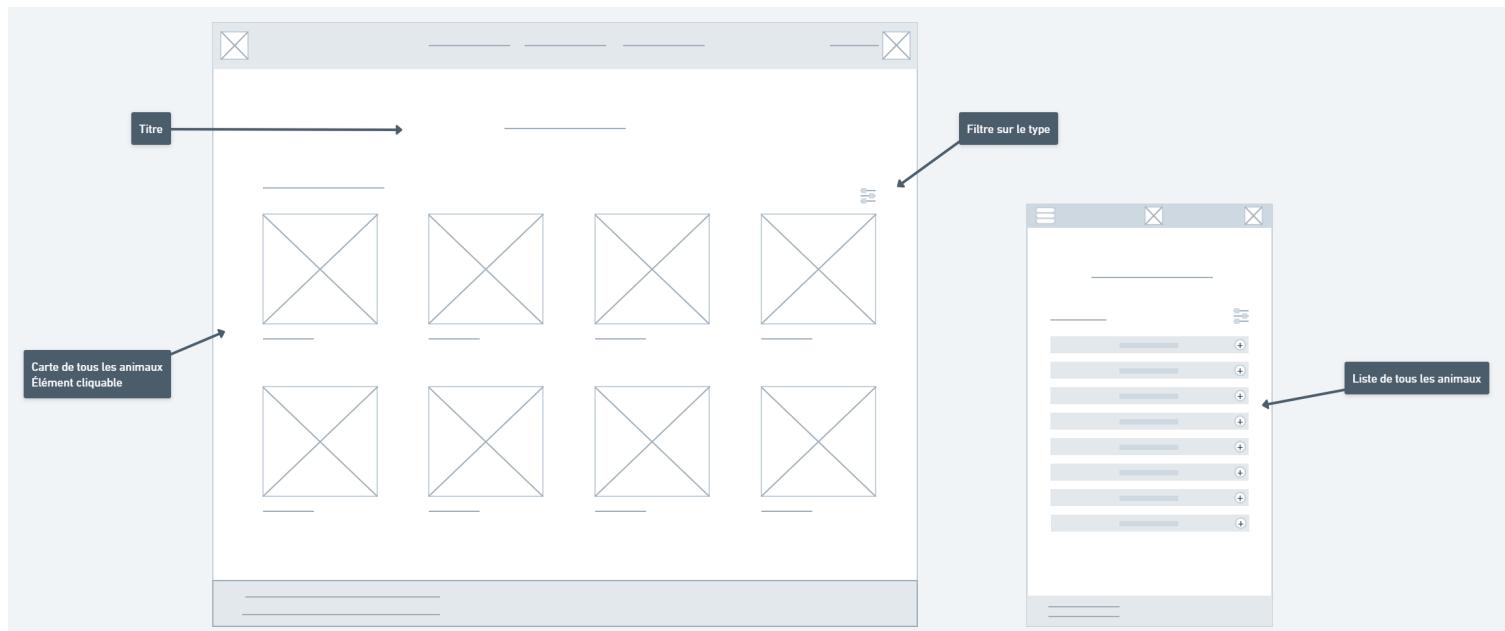
## Fiche chien



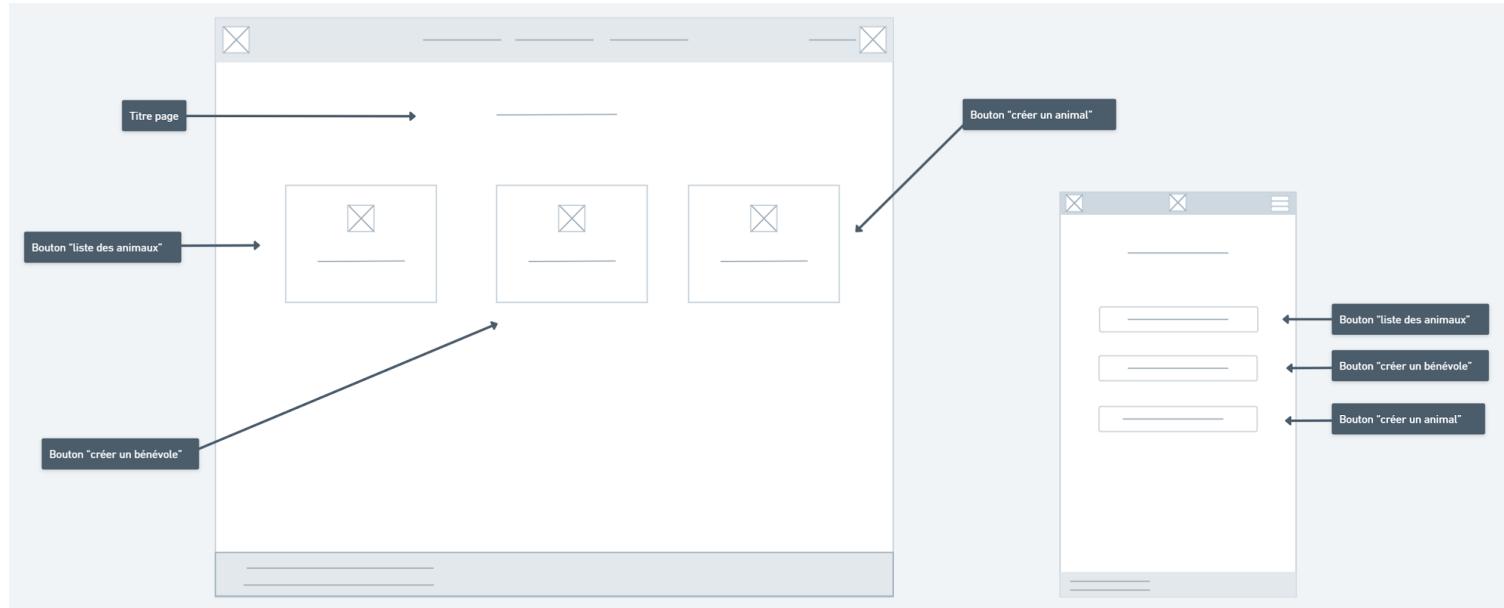
## Fiche chat



## Liste animaux



## Tableau de bord



## Création d'un animal

This wireframe diagram shows the creation form for an animal. The left side contains a vertical list of input fields: "Titre", "Selectionner espèce animal", "Nom animal", "Selectionner tags", "Race", "Age , poids", "Sexe", and "Numéro de cage". To the right of these fields is a circular "X" icon labeled "Photo de l'animal". Below the photo field is a large rectangular input area labeled "Informations sur l'animal". At the bottom of the form is a horizontal bar. To the right of the form is a sidebar with a scrollable list of input fields and a "Validation du formulaire" button. Arrows point from the sidebar validation button back to the main form's validation area.

## Création d'un bénévole

This wireframe shows the 'Création d'un bénévole' (Create Volunteer) screen. It features a header bar with a close button on the right. On the left, there's a 'Titre' input field with an arrow pointing to it. Below it is a dark box containing the text: 'Formulaire d'inscription du bénévole (prénom / nom / email / select du "expérience" de bénévole / mot de passe)'. In the center, there's a large input area with several horizontal lines for text input. To the right, a smaller mobile device icon displays a simplified version of the form.

## Profil

This wireframe shows the 'Profil' (Profile) screen. It has a header bar with a close button. On the left, there are three input fields: 'Titre' (with an arrow), 'photo de profil' (with an arrow), and 'Editer le profil' (with an arrow). The 'photo de profil' field contains a placeholder image icon. To the right, there's a large input area with several horizontal lines for text input. A dark box labeled 'Informations de profil (identifiant, mot de passe ...)' has an arrow pointing to the right side of the input area. A mobile device icon on the right shows a simplified profile view.

## VIII. Charte graphique et logo

L'application étant précisément à destination des bénévoles de la SPA, l'objectif était de s'approcher des codes de son identité visuelle afin de garder cohérence et harmonie. En analysant le site de cette dernière et en nous assurant des droits d'utilisation de la charte graphique, nous avons pu reprendre les différentes typologies et couleurs utilisées.

TYPOGRAPHY		COLORS	
<b>AMATIC SC</b>		#eb651c R235 G101 B28 A1	
REGULAR 400	BOLD 700	<b>Buttons &amp; Icons</b>	
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR	LOREM IPSUM DOLOR SIT AMET, CONSECTETUR	#222222 R34 G34 B34 A1	
<b>ROBOTO</b>		<b>Text</b>	
REGULAR 400	<b>BOLD 700</b>	#928161 R146 G129 B97 A1	
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do	<b>  Lorem ipsum dolor   sit amet, consectetur   adipiscing elit, sed</b>	<b>Icons 1</b>	
		#ffffaf R255 G250 B247 A1	#fdf3ed R253 G243 B237 A1
		<b>Primary</b>	<b>Secondary</b>
		#35450a R53 G69 B10 A1	
		<b>Icons 2</b>	

Pour le logo, nous avons fait le choix de garder un visuel sobre, épuré et moderne. Pour rester dans cette optique, la police comfortaa nous a semblé appropriée et le "O" a été transformé simplement en empreintes de chien pour faire le lien avec les animaux et la SPA.



## IX. Exemples de maquettes

toutpoils

App de gestion des actions de bénévolats



Email

Password

SE CONNECTER

Termines et conditions applicables

toutpoils



App de gestion des actions de bénévolats

Email

Password

SE CONNECTER

Termines et conditions applicables

toutpoils

[Accueil](#) [Sortir un chien](#) [Visiter la chatterie](#) [Voir tous les animaux](#) [DECONNEXION](#)

TITRE DE LA PAGE  
sous titre de la page

**Sortir un chien**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



**Visiter la chatterie**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Mentions Légales [Politique de confidentialité](#)

Copyright - 2023

toutpoils

**Sortir un chien**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



**Jouer avec les chats**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Mentions Légales [Politique de confidentialité](#)

Copyright - 2023

**tout poils**

Accueil [Sortir un chien](#) Visiter la chatterie Voir tous les animaux DECONNEXION

## TITRE DE LA PAGE

### sous titre de la page



Diego  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Amanda  
7 ans ♀  
Dernière sortie : 2 jours 11 heures



Rex  
4 ans ♂  
Dernière sortie : 2 jours 15 heures



Aiko  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Nina  
5 ans ♀  
Dernière sortie : 2 jours 11 heures

**Filtres**

Gabarit animal : Petit Moyen Grand

Sexe : Mâle Femelle

Age : 0 ans à 7 ans

Comportement : Sélectionner : Jouer Dynamique Rapide

**APPLIQUER FILTRES**

Mentions Légales Politique de confidentialité Copyright - 2023

**tout poils**

## TITRE DE LA PAGE

### sous titre de la page



Diego  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Amanda  
7 ans ♂  
Dernière sortie : 2 jours 15 heures



Aiko  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Nina  
5 ans ♂  
Dernière sortie : 2 jours 15 heures



Rex  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Biscuit  
7 ans ♂  
Dernière sortie : 2 jours 15 heures



Pope  
2 ans ♂  
Dernière sortie : 2 jours 15 heures



Chocolat  
3 ans ♂  
Dernière sortie : 2 jours 15 heures

**tout poils**

Accueil [Sortir un chien](#) Visiter la chatterie Voir tous les animaux DECONNEXION

## FICHE DE DIEGO

### sous titre de la page

[Retour à la liste](#)



4 ans Age  
Moyen Gabarit  
Mâle Sexe

Diego  
Jouer Dynamique Rapide

**CAGE N° 11**

Dernière sortie : 2 jours 15 heures

**BIOGRAPHIE**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

**DERNIERES BALADES**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**BONNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**MOYENNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**BONNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**MAUVAISE**

**CARGER HISTORIQUE BALADES**

Mentions Légales Politique de confidentialité Copyright - 2023

**tout poils**

## FICHE DE DIEGO

### sous titre de la page

[Retour à la liste](#)



4 ans Age  
Moyen Gabarit  
Mâle Sexe

Diego  
Jouer Dynamique Rapide

**CAGE N° 11**

Dernière sortie : 2 jours 15 heures

**BIOGRAPHIE**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

**DERNIERES BALADES**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**BONNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**MOYENNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**BONNE**

le 18/01/2023  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
**MAUVAISE**

**CARGER HISTORIQUE BALADES**

Mentions Légales Politique de confidentialité Copyright - 2023

# SPÉCIFICATIONS TECHNIQUES

Pour réaliser les spécifications techniques, nous avons tout d'abord réfléchi aux technologies que nous souhaitons utiliser pour réaliser ce projet. Puis nous nous sommes penchés sur quels navigateurs nous souhaitons que l'application soit utilisable de manière optimale.

Au vu du type d'application particulier, à savoir une application uniquement destinée à une antenne locale de la SPA, nous avons également réfléchi à la manière dont nous pourrions déployer cette application.

Enfin nous avons travaillé sur les documents liés à la réalisation de la base de données, puis à l'architecture de notre back-end et enfin aux routes de notre front-end.

## I. Technologies

- **Developer experience & gestionnaire de paquets**
  - Yarn
  - Eslint
  - Prettier
- **Front-end**
  - React
  - Bootstrap (react-bootstrap)
  - Sass
  - Redux
  - React-router
  - React Query et Axios
- **Back-end**
  - Express
  - Prisma
  - Joi pour la validation de données
  - Nodemailer pour l'envoi d'email, avec SendInBlue en fournisseur SMTP

## II. Navigateurs compatibles

Selon une étude de marché de 2022, les navigateurs les plus utilisés sont :

- **Sur desktop**
  - Chrome : 57,13 %
  - Firefox : 16,36 %
  - Edge : 11,74 %
- **Sur mobile**
  - Chrome : 56,03 %
  - Safari : 30,15 %

Notre application devra donc être compatible avec les versions les plus récentes des navigateurs cités ci-dessus.

Source:<https://www.leptidigital.fr/webmarketing/parts-de-marche-navigateurs-web-10814/#:~:text=R%C3%A9sum%C3%A9%20de%20la%20partition%20des%20navigateurs%20web%20en%20France%20sur,Edge%20%3A%2011%2C74%20%25>

### III. Possibilités de déploiement

Plusieurs possibilités de déploiement s'offrent au client (SPA), avec des solutions plus ou moins faciles d'utilisation, et plus ou moins coûteuses tant au niveau humain que financier. Nous avons étudié 3 cas :

#### 1. Le déploiement en local

Avantages	Inconvénients
<ul style="list-style-type: none"><li>• Peu cher, car nécessite seulement un ordinateur connecté au réseau local et allumé</li><li>• Possibilité de déployer l'ensemble de l'application assez rapidement (par exemple via Docker)</li><li>• Peu de risque de piratage</li><li>• Contrôle total sur les données</li><li>• Peu de données, utilisation d'un ORM totalement envisageable</li></ul>	<ul style="list-style-type: none"><li>• Uniquement utilisable si les devices des autres utilisateurs (mobiles, pc, tablettes) sont connectés au même réseau local</li><li>• Nécessite une IP statique sur le réseau</li><li>• Donne une url peu lisible (ex: <a href="http://192.168.0.13/animals">http://192.168.0.13/animals</a>)</li><li>• Nécessite une personne maîtrisant bien le déploiement de l'application. Par exemple, pour éditer les variables d'environnement (ex: serveur SMTP ou identifiants d'upload d'image)</li><li>• Processus de mise à jour de l'app à faire manuellement</li></ul>

**Estimation du coût mensuel :** quasiment nul

## 2. Le déploiement en cloud par l'antenne locale de la SPA

Avantages	Inconvénients
<ul style="list-style-type: none"> <li>• Contrôle total sur les données</li> <li>• Peu de données, utilisation d'un ORM envisageable</li> <li>• Utilisable partout</li> <li>• Les hébergeurs cloud fournissent généralement une URL accessible (ex: spa-crozon.vercelapp.com)</li> </ul>	<ul style="list-style-type: none"> <li>• Un peu plus cher que le déploiement en local, coût supporté par l'antenne locale de la SPA</li> <li>• Nécessite une personne maîtrisant bien le déploiement de l'application. Par exemple, pour éditer les variables d'environnement (ex: serveur SMTP ou identifiants d'upload d'image)</li> <li>• Processus de mise à jour de l'app à faire manuellement</li> </ul>

**Estimation du coût mensuel :**

Quantité	Nom	Prix
2	Heroku Basic Dyno - 1 front / 1 back	14
1	Heroku Basic PostgreSQL	9
1	Plugin Heroku Sendgrid - envoi email (serveur SMTP)	gratuit (jusqu'à 12.000 emails/mois)
1	Amazon S3 - stockage d'images	gratuit *

\* Estimation difficile, varie en fonction de l'usage

Gratuit pendant les 12 premiers mois ([présentation de AWS S3](#))

Tarifs: <https://aws.amazon.com/fr/s3/pricing/?p=pm&c=s3&z=4>

**TOTAL MINIMUM = 23€/mois**

### **3. Le déploiement en cloud centralisé par la SPA (SaaS - Software as a Service)**

Avantages	Inconvénients
<ul style="list-style-type: none"> <li>• Beaucoup plus simple d'utilisation</li> <li>• Coût assumé par le siège de la SPA</li> <li>• Utilisable partout</li> <li>• Nom de domaine plus clair, ex: benevoles.spa.fr)</li> <li>• Possibilité d'avoir un support technique plus performant pour les bénévoles sur le terrain</li> <li>• Facilité à déployer des mises à jour</li> <li>• Centralisation des données</li> </ul>	<ul style="list-style-type: none"> <li>• Beaucoup plus cher</li> <li>• Nécessite plusieurs personnes capables de gérer le déploiement et la scalabilité de l'infrastructure de l'app</li> <li>• Nécessiterait une réécriture de l'infrastructure de l'app</li> <li>• Le grand nombre de données, l'utilisation d'un ORM ralentirait l'application</li> </ul>

#### **Estimation du coût mensuel:**

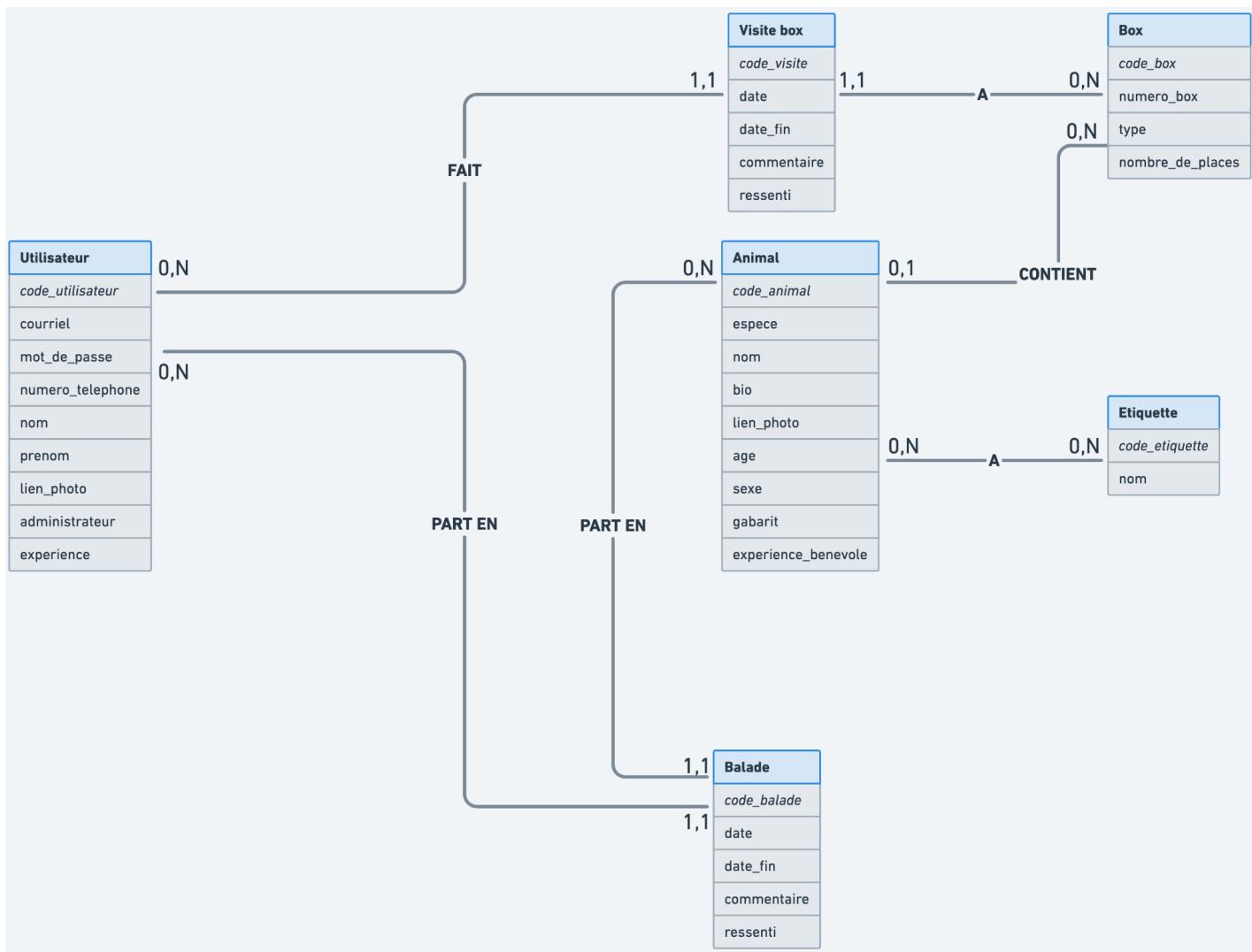
Il est difficile de fournir une estimation du prix, car l'application pourrait être utilisée par plusieurs milliers de visiteurs uniques mensuels. De plus, le coût de stockage des données devrait être élevé, la base de données devant également contenir l'ensemble des données des animaux, des promenades, des visites, des box de toutes les antennes locales des SPA, ce qui représentent plusieurs milliers d'entrées.

On peut également imaginer que le coût du stockage d'images par le biais d'un fournisseur de stockage comme le service Amazon S3 serait bien plus élevé, car facturé à la quantité stockée mais également au nombre de requêtes (lecture, modifications, etc) sur les images.

Au vue de cette étude, le déploiement en cloud par l'antenne locale de la SPA nous semble le plus approprié dans un premier temps. En effet, il donne un bon compromis entre accès à l'application (utilisation partout) et le coût. Le déploiement en cloud centralisé par la SPA (SaaS - Software as a Service) pourrait être envisagé dans un second temps si l'application est utilisée par beaucoup de personnes mais nécessiterait une restructuration du code.

## IV. Création de la base de données

### A. MCD



### B. MLD

**user** (id, email, password, name, firstname, phone\_number, url\_image, admin, experience)  
**box** (id, type, number, nb\_of\_places)  
**animal** (id, species, name, url\_image, age, bio, gender, size, volunteer\_experience, #**box\_id**)  
**walk** (id, date, end\_date, comment, feeling, #**user\_id**, #**animal\_id**)  
**tag** (id, name)  
**animal\_has\_tag** (#**animal\_id**, #**tag\_id**)  
**visit** (id, date, end\_date, comment, feeling, #**user\_id**, #**box\_id**)

### C. Dictionnaire des données

Nom de la donnée	Désignation	Type	contrainte
code_utilisateur	Code numérique d'un utilisateur	integer	generated as identity <b>primary key</b>
courriel	Adresse email d'un utilisateur	text	not null
mot_de_passe	Mot de passe d'un utilisateur	text	not null
numéro_telephone	Numéro de téléphone d'un utilisateur	text	
nom	Nom de l'utilisateur	text	not null
prenom	Prénom de l'utilisateur	text	not null
lien_photo	Url vers la photo de l'utilisateur	text	
administrateur	Confirmation du statut admin de l'utilisateur	boolean	not null, default false
experience	Niveau d'expérience de l'utilisateur	text	not null check ("beginner", "medium", "expert"), default ("beginner")
code_animal	Code numérique d'un animal	integer	generated as identity <b>primary key</b>
espece	Espèce de l'animal	text	not null check("cat", "dog", "other")
nom	Nom de l'animal	text	not null
bio	Résumé de l'animal	text	
lien_photo	Url vers la photo de l'animal	text	
etiquette	Etiquette de la caractéristique de l'animal	text	not null
age	Age de l'animal	timestamptz	
sexe	Sexe de l'animal	text	check ("male", "female")
gabarit	Gabarit de l'animal	text	check ("small", "medium", "big")
Expérience_benevole	Expérience demandée pour s'occuper de l'animal	text	not null default 'beginner'
code_etiquette	Code numérique de l'étiquette	integer	generated as identity <b>primary key</b>
nom	Nom de l'étiquette	text	not null
code_balade	Code numérique de la balade	integer	generated as identity <b>primary key</b>
date	Date d'une balade	timestamptz	not null
commentaire	Commentaire sur la balade	text	
ressenti	Ressenti avec l'animal lors de la balade	text	not null check ("bad", "medium", "big") default ("good")

<b>code_visite</b>	Code numérique de la visite	integer	generated as identity <b>primary key</b>
date	Date de la visite	timestamptz	not null
commentaire	Commentaire sur la visite	text	
<b>code_box</b>	Code numérique d'un box	integer	generated as identity <b>primary key</b>
numero_box	Numéro du box	alphanumérique	not null
type	Type du box	text	not null , check ("chien","chat","autre") default ("other")
nombre_de_place	Nombre de place possible dans un box	integer	not null , check ("chien","chat","autre") default (1)

## V. Routes front et back

### A. Back-end

	ENDPOINT	OUTPUT	CONTROLLER	AUTH
<b>Connexion</b>				
POST	/auth/login	Permet de se connecter	authController.login	Non-connecté
<b>users</b>				
GET	/users	Retourne la liste des utilisateurs	usersController.getAllUsers	Connecté
POST	/users	Utilisateur créé - on retire le mot de passe de l'objet retourné	usersController.create	Administrateur
GET	/users/:id	retourne un utilisateur	usersController.getOne	Connecté
PATCH	/users/:id	met à jour un utilisateur puis retourne l'utilisateur modifié	usersController.update	Connecté
DELETE	/users/:id	ne retourne rien (juste un status HTTP 204)	usersController.delete	Connecté
<b>animals</b>				
GET	/animals	retourne une liste d'animaux	animalsController.getAll	Connecté
filtre	/animals?gabarits=GROS	retourne une liste d'animaux avec un filtre qui retourne les gros gabarits		
POST	/animals	Retourne l'animal créé	animalsController.create	Administrateur
GET	/animals/:id	Retour l'animal sélectionné	animalsController.getOne	Connecté
GET	/animals/:id/walks	Récupère la liste des balades d'un animal en particulier	animalsController.getWalksOfAnimal	Connecté
PATCH	/animals/:id	Met à jour les informations d'un animal en particulier	animalsController.update	Administrateur
DELETE	/animals/:id	ne retourne rien (juste un status HTTP 204)	animalsController.delete	Administrateur
<b>walks</b>				
GET	/walks	Récupère la liste de toutes les balades	walksController.getAll	Connecté
filtre	/walks?animal_id=:id	exemple: récupère la liste de toutes les balades d'un animal en particulier		
POST	/walks	Crée une nouvelle balade	walksController.create	Connecté
PATCH	/walks/:id	Met à jour les informations d'une balade en particulier	walksController.update	Connecté
GET	/walks/:id	Récupère les détails d'une balade en particulier	walksController.getOne	Connecté
DELETE	/walks/:id	ne retourne rien (juste un status	animalsController.delete	Administrateur

		HTTP 204)		
<b>boxes</b>				
GET	/boxes	Récupère la liste de toutes les boxes	boxesController.getAll	Connecté
filtré	/boxes?type=:CHAT	exemple: récupère la liste des boxes contenant des chats		
POST	/boxes	Crée un nouveau box	boxesController.create	Administrateur
PATCH	/boxes/:id	Met à jour les informations d'un box en particulier	boxesController.update	Administrateur
GET	/boxes/:id	Récupère un box	boxesController.getOne	Connecté
GET	/boxes/:id/animals	Retourne les animaux contenus dans un box	boxesController.getAnimals	Connecté
GET	/boxes/:id/visits	Retourne la liste des visites d'un box	boxesController.getVisitsOfOneBox	Connecté
DELETE	/animals/:id	ne retourne rien (juste un status HTTP 204)	animalsController.delete	Administrateur
<b>visites</b>				
GET	/visits	Récupère la liste de toutes les visites	visitsController.getAll	Connecté
filtré	/visits?box=:id	exemple:récupère la liste des visites d'une boîte en particulier		
POST	/visits	crée une nouvelle visite	visitsController.create	Connecté
PATCH	/visits/:id	Met à jour les informations d'une visite en particulier	visitsController.update	Connecté
GET	/visits/:id	Récupère une visite	visitsController.getOne	Connecté
DELETE	/visits/:id	ne retourne rien (juste un status HTTP 204)	visitsController.delete	Connecté

## B. Front-end

ROUTE	PAGE	USER
<b>Connexion</b>		
/login	Page de connexion	Non-connecté
<b>Menu Home</b>		
/home	Page home	Bénévole
/walkingdog	Page liste des chiens à sortir	Bénévole
/visits	Page des boxes	Bénévole
/clean	Page nettoyages des boxes	Bénévole V2
/events	Page des événements	Bénévole V2
<b>Menu Dashboard</b>		
/admin	Page dashboard	Admin
/admin/create/user	Page inscription bénévole	Admin
/admin/create/card	Page création d'une fiche animal	Admin

/admin/users	Page liste des utilisateurs	Admin
<b>Pages en commun</b>		
/animals	Page animaux	Bénévole et admin
/animal/:animalId	Page d'un animal	Bénévole et admin
/box/:id	Page d'un box	Bénévole et admin
/profile	Page profil	Bénévole et admin

# RÉALISATIONS PERSONNELLES

Au cours de ce projet, j'ai eu l'occasion de travailler sur plusieurs domaines. J'ai évidemment travaillé sur les parties back et front de l'application, mais en tant que Git Master, j'ai eu l'occasion de découvrir la gestion de projet avec Git. Je me suis également occupé du déploiement de notre MVP, ainsi que de la partie ayant attrait à l'upload et l'hébergement d'images. Pour ce faire, j'ai appris à manier l'infrastructure AWS avec notamment le service S3.

Cependant deux points ont retenu mon attention lors de la réalisation du projet: le système d'upload d'images, ainsi que le composant react permettant le départ en balade d'un chien.

## I. Upload d'image

Le périmètre de l'envoie d'image couvre le front et le back. Pour expliquer la façon dont j'ai réalisé cette fonctionnalité, je vais en premier lieu présenter la gestion de l'envoi d'image au niveau du back, puis la façon dont je construis ma requête d'envoi d'image au niveau du front.

### A. Back

Pour réaliser cette fonctionnalité, j'ai utilisé le paquet [Multer](#). Multer est un middleware qui permet de traiter l'envoi d'image avec Express.

Concernant l'hébergement des images, j'ai choisi d'utiliser le service [S3](#) d'Amazon AWS. Pour interagir avec ce service, j'ai utilisé le [SDK](#) (Software Development Kit) d'AWS, et plus particulièrement les outils liés au service S3.

Tout d'abord, j'ai créé un fichier permettant la création de la connexion au service S3, avec les identifiants de connexion stockés dans le fichier .env.

```
1 import { S3Client } from '@aws-sdk/client-s3';
2
3 // Création d'un client S3 pour l'envoi des fichiers
4 const s3Client = new S3Client({
5   region: process.env.S3_REGION,
6   credentials: {
7     accessKeyId: process.env.S3_ACCESS_KEY,
8     secretAccessKey: process.env.S3_ACCESS_SECRET,
9   },
10 });
11
12 export default s3Client;
```

Puis j'ai créé un service permettant l'envoi des fichiers sur les buckets S3. Les buckets sont des sortes de containers de stockage que nos fichiers, qui permettent de les classer en fonction de l'utilisation de ces fichiers.

Pour notre projet, j'ai choisi de créer deux buckets:

- le bucket **animals**: contient les photos des animaux
- le bucket **users**: contient les photos de profil des utilisateurs

Au sein du service, j'ai créé une simple fonction qui prend en paramètre le nom du bucket de destination, et qui retourne un objet contenant les informations du bucket ciblé, retournant son nom ainsi que son URL public qui permettra de construire l'URL de l'image qui sera stocké en base de données.

```
9  const bucketInfo = (name) => {
10    if (name) {
11      switch (name) {
12        case 'animals':
13          return {
14            bucket: process.env.S3_ANIMAL_PICTURES_BUCKET_NAME,
15            publicUrl: `https://${process.env.S3_ANIMAL_PICTURES_BUCKET_NAME}.s3.${process.env.S3_REGION}.amazonaws.com/`,
16          };
17        case 'users':
18          return {
19            bucket: process.env.S3_USER_PICTURES_BUCKET_NAME,
20            publicUrl: `https://${process.env.S3_USER_PICTURES_BUCKET_NAME}.s3.${process.env.S3_REGION}.amazonaws.com/`,
21          };
22        default:
23          throw new Error('INVALID_UPLOAD_DESTINATION');
24      }
25    } else {
26      throw new Error('INVALID_UPLOAD_DESTINATION');
27    }
28  };
```

Puis, j'ai créé le service en lui-même qui sera exporté pour permettre à nos controller de procéder à l'upload d'image sur notre espace S3.

```
31  const uploadService = {
32    upload: async (destination, fileExtension, body) => {
33      if (!destination || !fileExtension || !body) {
34        throw new Error('MISSING_UPLOAD_PARAMETER');
35      }
36      try {
37        // On récupère les infos du Bucket AWS S3 ou l'on souhaite uploader l'image
38        const { bucket, publicUrl } = bucketInfo(destination);
39
40        // On génère un nom aléatoire, se terminant par l'extension du fichier
41        const name = `${crypto.randomUUID()}.${fileExtension}`;
42
43        // On envoie l'image vers s3, les données de l'image sont contenues dans "Body"
44        await s3Client.send(
45          new PutObjectCommand({
46            Region: process.env.REGION,
47            Bucket: bucket,
48            Key: name,
49            Body: body,
50          })
51        );
52
53        // On retourne le lien de l'image
54        return publicUrl + name;
55      } catch (error) {
56        throw new APIError({
57          message: 'UPLOAD_FAILED',
58          error,
59        });
60      }
61    },
62  };
```

On remarque que notre service exporte une méthode `upload` qui prend en paramètre le nom du bucket de destination de l'image, l'extension de l'image ainsi que le contenu de l'image. On voit dans cet extrait de code que la fonction `bucketInfo` définie précédemment est appelée pour retourner les informations du bucket visé. Ces informations sont utilisées par le SDK d'AWS avec la méthode `send`. Ici l'élément `s3Client` fait référence au client S3 que j'ai créé précédemment.

Enfin, cette fonction retourne l'URL de l'image stockée sur les serveurs S3.

Au niveau des controllers, j'importe donc mon service et je l'utilise en passant en paramètre le bucket de destination (ici **animals**), je passe l'extension que je récupère en séparant le nom du fichier par les `.`, avec la méthode `split('.')` et je récupère le dernier élément du tableau avec la méthode `pop()`. Et enfin, je passe le contenu de l'image qui est passé au niveau de la requête avec `req.file.buffer`

```
238 // Gestion de l'upload d'image
239 if (req.file) {
240     // on récupère l'extension du fichier
241     const fileExtension = req.file.originalname.split('.').pop();
242
243     // on fait appel au service d'upload
244     // le premier paramètre est le nom du bucket S3 dans lequel on veut stocker l'image
245     // le deuxième paramètre est l'extension de l'image
246     // le troisième paramètre est le contenu de l'image
247     const urlImage = await uploadService.upload(
248         'animals',
249         fileExtension,
250         req.file.buffer
251     );
252
253     // une fois l'image uploadée, on ajoute l'url de l'image dans l'objet animal, à sauvegarder en bdd
254     animal.url_image = urlImage;
255 }
```

On remarque ici l'utilisation de `req.file`. Cet objet est passé dans la requête par le middleware de gestion des upload d'images, qui utilise **multer**.

Pour créer le middleware `fileUpload`, qui traite les requêtes contenant des images, j'ai créé une surcouche de multer pour éviter de devoir pour chaque route passer un objet de configuration de multer.

```
1 import multer from 'multer';
2 import APIError from '../services/APIError.service.js';
3
4 // On définit qu'on souhaite que le fichier reste dans la mémoire vive du serveur et qu'il ne soit pas enregistré dans un dossier de notre application
5 const storage = multer.memoryStorage();
6
7 const fileFilter = (req, file, cb) => {
8   /**
9    * On vérifie que le fichier envoyé est bien une image
10   */
11  if (
12    file.mimetype === 'image/gif' ||
13    file.mimetype === 'image/png' ||
14    file.mimetype === 'image/jpeg' ||
15    file.mimetype === 'image/webp'
16  ) {
17    // on passe "true" pour signifier que le fichier est valide, et passer à l'étape suivante de la requête
18    cb(null, true);
19  } else {
20    // sinon on passe une erreur
21    cb(
22      new APIError({
23        code: 400,
24        message: 'INVALID_PICTURE_FORMAT',
25      })
26    );
27  }
28};
29
30 // On exporte le middleware, qui est une "surcouche" de multer
31 // auquel on ajoute les options de configuration (fileFilter)
32 // et les options de stockage (storage, ici stockée en mémoire)
33 const fileUpload = multer({
34   fileFilter,
35   storage,
36 });
37
38 export default fileUpload;
```

Enfin, au niveau des routes où je souhaite pouvoir permettre l'upload fichier, j'utilise ce middleware.

```
57 /* je veux créer un nouvel animal
58 */
59 animalRouter.post(
60   '/',
61   authentication,
62   // on ajoute le middleware qui va gérer l'upload du fichier
63   // on précise que le fichier sera envoyé dans le champ "image" du formulaire, et qu'il sera unique
64   fileUpload.single('image'),
65   validate(animalsValidation.create, 'body'),
66   animalsController.create
67 );
```

On remarque ici que le middleware utilise la méthode `single` de Multer pour indiquer à express qu'on souhaite que le formulaire contienne uniquement un seul fichier, et que ce fichier doit être contenu dans l'élément “image” du formulaire.

Le middleware se charge alors de stocker l'image envoyée en mémoire vive, puis vérifie que le fichier est bien un fichier image avec les extensions de fichier autorisées, puis continue la requête en attachant l'élément `req.file`.

## B. Front

La mise en place de l'upload au niveau du front reste assez simple, j'ai simplement ajouté un champ de type `file` dans mon formulaire de création d'un animal.

```
293 <Form.Group className='mb-3'>
294     <Form.Label className='creation-form-label'>
295         Image profil
296     </Form.Label>
297     <Form.Control type='file' {...register('image')} />
298 </Form.Group>
```

Puis une fois que l'utilisateur valide le formulaire, je vérifie que le formulaire contient une image. Si c'est le cas, je crée un nouveau `FormData`, pour envoyer un formulaire à la place d'une requête contenant du JSON. A ce formulaire, j'attache l'image de du champ défini précédemment ainsi que toutes les autres informations que je souhaite passer au back. Puis j'exécute la requête.

```
80  /**
81  * VALIDATION DU FORMULAIRE
82  */
83 const onSubmitHandler = (data) => {
84     delete data.tags;
85
86     let body;
87     /**
88      * En cas d'upload'image
89      */
90     if (data.image && data.image.length > 0) {
91         /**
92             * On crée un objet FormData pour envoyer les données, qui simule un formulaire
93             * C'est obligatoire pour envoyer des fichiers.
94             */
95         body = new FormData();
96         // on ajoute l'image au FormData, issue de l'input type="file" du formulaire
97         body.append('image', data.image[0]);
98         // on supprime l'image du state, pour ne pas l'envoyer en double
99         delete data.image;
100
101        /**
102            * On ajoute les tags au FormData
103            */
104        if (tags && tags.length > 0) {
105            body.append('tags', tags.join(','));
106        }
107
108        for (let key in data) {
109            body.append(key, data[key]);
110        }
111
112        // on ajoute le box_id au FormData
113        body.append('box_id', Number(selectedBox));
114    } else {
115        // Si pas d'upload d'image, on envoie un objet classique au format JSON
116        body = { ...data, tags: tags, box_id: Number(selectedBox) };
117    }
118
119    api.post('/animals', body).then(function (response) {
120        if (response.status === 201) {
121            // si la création a réussi, on redirige vers le dashboard
122            navigate('/admin');
123        }
124    });
125};
```

## II. Bouton de départ en balade

La création de ce bouton me paraissait assez simple dans la théorie, je devais simplement vérifier que l'utilisateur avait le droit de partir en balade, que l'animal n'était pas sorti dans la journée et gérer l'affichage d'un bouton actif "partir en balade", ou "terminer la balade", ou un bouton inactif "cet animal est déjà sorti aujourd'hui".

En réalité, ce bouton devait gérer plusieurs états et conditions:

- Le bouton ne doit pas être visible pour un administrateur, les administrateurs n'ayant pas le droit de partir en balade
- Le bouton doit être inactif si l'animal est déjà sorti dans la journée
- Le bouton doit être inactif si l'animal est déjà en cours de balade.
- Si c'est l'utilisateur connecté qui est en balade avec l'animal, il doit pouvoir arrêter la balade
- La durée maximale d'une balade étant de une heure, si le chien est en balade depuis plus d'une heure, le bouton doit afficher "ce chien est déjà sorti aujourd'hui", même si l'utilisateur qui promenait le chien n'a pas cliqué sur "terminer la balade".

J'ai donc créé un composant *StartWalkButton*, prenant en paramètre un objet contenant les informations de l'animal, notamment ces dernières balades (*animal.walks*).

```
16 const StartWalkButton = ({ animal }) => {
17   // on récupère les informations de l'utilisateur connecté
18   const { id, experience, admin } = useSelector((state) => state.loginSettings);
19   // contiendra les informations de la balade créée en BDD lors du départ en balade
20   const [startedWalk, setStartedWalk] = useState(undefined);
21   // Gère l'affichage de la modale de confirmation de départ en balade
22   const [showModal, setShowModal] = useState(false);
23   // Gère l'affichage de l'éditeur de balade en fin de balade
24   const [showEndEditor, setShowEndEditor] = useState(false);
25
26   // On récupère la dernière balade de l'animal, ou null si aucune balade n'a été enregistrée
27   const [lastWalk, setLastWalk] = useState(
28     animal.walks[animal.walks.length - 1] ?? null
29   );
30
31   // Requête de création de balade, géré par react-query
32   const {
33     isLoading,
34     mutate: startWalk,
35     error,
36   } = useMutation({
37     mutationFn: (data) => {
38       return walksRequest.create(id, data.animal_id);
39     },
40     onSuccess: (data) => {
41       setStartedWalk(data.data);
42       setLastWalk(data.data);
43       setShowModal(false);
44     },
45   });
46
47 //...
```

Au niveau de la logique du rendu du bouton, j'encapsule la logique du bouton dans une condition qui vérifie si l'animal est bien passé en paramètre, et si l'utilisateur n'est pas un administrateur, auquel cas je ne retourne rien et le composant n'affiche rien.

```

49 // on vérifie que l'animal existe et que l'utilisateur n'est pas un admin
50 // Pour rappel: les admins ne peuvent pas partir en balade avec les animaux
51 if (animal && !admin) {
52 /**
53 * durée de la balade max 1h donc on check si la dernière sortie remonte à plus d'une heure, pour déterminer si l'animal est en cours de balade
54 * On vérifie également que l'animal n'est pas déjà sorti dans la journée
55 */
56
57 if (
58     lastWalk === null ||
59     (DateTime.fromISO(lastWalk?.date).plus({ hour: 1 }) <= DateTime.now() &&
60     DateTime.fromISO(lastWalk?.date) <= DateTime.now().startOf('day'))
61 ) {
62     // on vérifie que l'utilisateur le niveau d'expérience requis pour partir en balade avec l'animal
63     // On utilise la fonction experienceConverter du fichier experience.utils.js, pour convertir l'expérience de l'utilisateur de string à number
64     if (
65         experienceUtil.experienceConverter(experience) >=
66         experienceUtil.experienceConverter(animal.volunteer_experience)
67     ) {
68         return (
69             <Button
70                 onClick={() => setShowModal(true)}
71                 role='button'
72                 tabIndex='0'
73                 variant='primary'
74                 lg={2}
75             >
76             Partir en balade
77         </Button>
78
79         /* Modale de confirmation de la balade */
80
81         <Modal show={showModal} onHide={() => setShowModal(false)}>
82             <Modal.Header closeButton>
83                 <Modal.Title>Confirmation de balade</Modal.Title>
84             </Modal.Header>
85             <Modal.Body>
86                 Etes vous sur de vouloir sortir cet animal ?
87             </Modal.Body>
88             <Modal.Footer>
89                 <div className='d-flex'>
90                     {isLoading ? (
91                         <Button className='me-2' variant='primary' disabled>
92                             Chargement...
93                         </Button>
94                     ) : (
95                         <Button
96                             className='me-2'
97                             variant='primary'
98                             onClick={() => {
99                                 startWalk({ animal_id: animal.id });
100                            }}
101                         >
102                         Oui
103                         </Button>
104                     )}
105                     {error && (
106                         <div style={buttonStyle}>
107                             Une erreur est survenue, merci de retenter plus tard
108                         </div>
109                     )}
110                     <Button
111                         variant='secondary'
112                         onClick={() => setShowModal(false)}
113                     >
114                         Annuler
115                     </Button>
116                 </div>
117             </Modal.Footer>
118         </Modal>
119     );
120 }
121 );
122 } else {
123     return (
124         <div style={buttonStyle}>
125             Vous ne pouvez pas partir en balade (expérience insuffisante)
126         </div>
127     );
128 }
129 //...
130

```

Dans un premier temps on remarque que ma première condition consiste à vérifier que l'animal n'a aucune balade OU n'est pas déjà sorti depuis minuit ET qu'il n'est pas en balade. Ensuite, je vérifie que l'utilisateur a le niveau nécessaire pour sortir le chien. Pour ce faire, les niveaux étant stockés en base de données en format textuel (beginner, medium, expert), je

convertis le niveau en nombre pour pouvoir effectuer une comparaison logique grâce à la fonction *experienceConverter*.

```
15  experienceConverter: (level) => {
16      switch (level) {
17          case 'MEDIUM':
18              return 1;
19          case 'EXPERT':
20              return 2;
21          default: // retourne par défaut un niveau débutant (BEGINNER=0)
22              return 0;
23      }
24 },
```

Si le niveau de l'utilisateur est supérieur ou égal au niveau requis pour sortir l'animal, alors j'affiche le bouton pour partir en balade.

Le composant affiche un simple bouton, qui, une fois cliqué affiche une modale de confirmation de départ en balade. Si l'utilisateur valide le départ en balade, la fonction *startWalk* déclenche la requête pour enregistrer la nouvelle balade en base de données.

Une fois la requête effectuée, je remplace la dernière balade enregistrée dans le stade du composant par la nouvelle balade enregistrée pour permettre à l'utilisateur de voir le bouton "terminer la balade" sans avoir à recharger la page.

Enfin après avoir géré les conditions pour l'affichage du bouton de départ en balade, je gère l'état inverse à savoir afficher un bouton "terminer la balade", si c'est l'utilisateur qui est parti en balade avec l'animal ou l'affichage d'un bouton désactivé indiquant que l'animal est déjà en balade ou qu'il est déjà sorti durant la journée.

```

130    //...
131  else {
132    /**
133     * SI LE CHIEN EST EN COURS DE BALADE
134     * on récupère la dernière visite enregistrée
135     * on vérifie que l'ID de l'utilisateur qui a créé la dernière balade correspond à l'utilisateur connecté
136     */
137   if (
138     startedWalk ||
139     (lastWalk.user_id == id &&
140      lastWalk.end_date == undefined &&
141      DateTime.fromISO(lastWalk.date).plus({ hour: 1 }) >= DateTime.now())
142   ) {
143     return (
144       <>
145       <WalkEditor
146         walk={startedWalk || lastWalk}
147         show={showEndEditor}
148         endingWalk={true}
149         onClose={() => setShowEndEditor(false)}
150         onUpdate={(data) => {
151           setStartedWalk(undefined);
152           setLastWalk(data);
153         }}
154       />
155       <Button
156         variant='primary'
157         onClick={() => {
158           setShowEndEditor(true);
159         }}
160         role='button'
161         tabIndex='0'
162         className='is-walking'
163       >
164         Terminer la balade
165       </Button>
166     </>
167   );
168 } else {
169   // sinon on affiche que l'animal est déjà sorti, ou en cours de sortie avec un autre utilisateur
170   if (
171     DateTime.fromISO(lastWalk.date).plus({ hour: 1 }) >= DateTime.now() &&
172     lastWalk.end_date == undefined
173   ) {
174     return (
175       <p className='bg-secondary p-2 text-light'>
176         Cet animal est en cours de sortie
177       </p>
178     );
179   } else {
180     return (
181       <p className='bg-secondary p-2 text-light'>
182         Cet animal est déjà sorti aujourd'hui
183       </p>
184     );
185   }
186 }
187 }
188 }
```

Sur le plan théorique, la réalisation semblait simple mais c'était la gestion des différents états du bouton qui rendait la réalisation de ce composant difficile. L'autre difficulté résidait dans la gestion de l'heure, notamment pour savoir si le chien était déjà sorti dans la journée. Pour résoudre cette difficulté, j'ai utilisé le module [luxon](#) qui permet de manipuler simplement les dates, et notamment avec la méthode `startOf()` qui permet de déterminer le timestamp du début d'une heure, d'une journée, d'un mois etc...

Luxon permet également de faire des comparaisons avec les dates plus facilement qu'en javascript natif.

## PRÉSENTATION DU JEU D'ESSAI

Pour illustrer la création d'un animal avec une image téléversée depuis l'ordinateur du client, voici un jeu d'essai qui permet de vérifier le bon fonctionnement de la requête.

Pour réaliser ce jeu d'essai, je vais me servir du logiciel [Postman](#) qui permet de tester, et de documenter les requêtes vers une API.

Dans un premier temps, j'observe l'état actuel de notre base de données, avant l'ajout de l'animal. La table qui sera mise à jour lors de la création d'un animal est la table "Animal".

pgAdmin 4

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.Animal/t... public.Animal/toutopois\_db/postgres@LOC

Extension Foreign D Language Publication Schemas Data Output Messages Notifications

public

	id [PK] integer	species text	name text	gender text	age timestamp with time zone	size text	volunteer_experience text	box_id integer	url_image text
1	27	31	Zippo	MALE	2011-03-16 00:00:00+01	SMALL	EXPERT	9	[null]
2	FTS 28	32	Chanel	FEMALE	2020-08-06 00:00:00+02	BIG	MEDIUM	8	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/61482/sier">https://www.la-spa.fr/app/assets-spa/uploads/animals/61482/sier</a>
3	FTS 29	34	Pluto	MALE	2020-01-31 13:24:53+01	BIG	BEGINNER	7	[null]
4	FTS 30	35	Elvira	FEMALE	2020-08-10 00:00:00+02	BIG	EXPERT	11	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/68558/cay">https://www.la-spa.fr/app/assets-spa/uploads/animals/68558/cay</a>
5	FTS 31	36	Cactus	MALE	2021-01-19 00:00:00+01	MEDIUM	MEDIUM	8	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/20554/cac">https://www.la-spa.fr/app/assets-spa/uploads/animals/20554/cac</a>
6	FTS 32	37	Rox	MALE	2014-06-01 00:00:00+02	BIG	MEDIUM	9	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/27751/rost">https://www.la-spa.fr/app/assets-spa/uploads/animals/27751/rost</a>
7	FTS 33	38	Diego	MALE	2018-04-04 00:00:00+02	MEDIUM	BEGINNER	9	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/59614/sco">https://www.la-spa.fr/app/assets-spa/uploads/animals/59614/sco</a>
8	FTS 34	39	Rita	FEMALE	2008-01-03 00:00:00+01	BIG	BEGINNER	10	[null]
9	FTS 35	40	Wendy	FEMALE	2009-09-10 12:24:53+02	MEDIUM	BEGINNER	11	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/bf2bace-0">https://animalspictures.s3.eu-west-3.amazonaws.com/bf2bace-0</a>
10	FTS 36	41	Whisky	MALE	2020-01-26 00:00:00+01	MEDIUM	BEGINNER	7	<a href="https://i.pinimg.com/736x/5a/5f/b3/5a5fb3571e8614faafe9f8198">https://i.pinimg.com/736x/5a/5f/b3/5a5fb3571e8614faafe9f8198</a>
11	FTS 37	42	Floup	MALE	2022-01-01 00:00:00+01	MEDIUM	BEGINNER	11	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/50277/flou">https://www.la-spa.fr/app/assets-spa/uploads/animals/50277/flou</a>
12	FTS 38	44	Kimbo	MALE	2014-12-22 00:00:00+01	BIG	MEDIUM	13	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/7d01f1e3-6">https://animalspictures.s3.eu-west-3.amazonaws.com/7d01f1e3-6</a>
13	FTS 39	46	Milou	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	8	<a href="https://upload.wikimedia.org/wikipedia/commons/e/e7/Fox_Terrier">https://upload.wikimedia.org/wikipedia/commons/e/e7/Fox_Terrier</a>
14	FTS 40	47	Bill	MALE	2016-04-01 00:00:00+02	BIG	BEGINNER	15	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/385fb890-d">https://animalspictures.s3.eu-west-3.amazonaws.com/385fb890-d</a>
15	FTS 41	48	Kubo	MALE	2022-09-01 00:00:00+02	MEDIUM	BEGINNER	14	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/662c76c7-e">https://animalspictures.s3.eu-west-3.amazonaws.com/662c76c7-e</a>
16	FTS 42	49	Mika	MALE	2016-01-01 00:00:00+01	MEDIUM	MEDIUM	16	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/e647967e-6">https://animalspictures.s3.eu-west-3.amazonaws.com/e647967e-6</a>
17	FTS 43	50	Hobbo	MALE	2005-12-20 00:00:00+01	BIG	EXPERT	17	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/2b19f0f8-f">https://animalspictures.s3.eu-west-3.amazonaws.com/2b19f0f8-f</a>
18	FTS 44	51	Pistache	MALE	2014-04-15 00:00:00+02	SMALL	EXPERT	14	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/5182ef48-e">https://animalspictures.s3.eu-west-3.amazonaws.com/5182ef48-e</a>
19	FTS 45	52	O'titi	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	15	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/eab0cf60-b">https://animalspictures.s3.eu-west-3.amazonaws.com/eab0cf60-b</a>
20	FTS 46	53	bambou	FEMALE	2023-03-01 00:00:00+01	SMALL	BEGINNER	6	[null]
21	FTS 47	56	malo	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	1	[null]

La requête de création d'un animal n'étant autorisée que pour les administrateurs, je dois tout d'abord récupérer un JWT que je vais ensuite passer à ma requête de création d'un animal.

J'effectue donc une requête vers l'endpoint de récupération du token, avec mes identifiants administrateur.

POST http://localhost:3005/v1/auth/login

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

Body (Pretty, Raw, Preview, Visualize, JSON)

1 {  
2 ... "email": "admin@toutopoils.fr",  
3 ... "password": "XXXXXXXXXXXXXX"  
4 }

Status: 200 OK Time: 137 ms Size: 460 B Save Response

1 {"token": "eyJhbGciOiJIUzI1NiJ9.eyJpZCI6MjcsImFkbWluIjp0cnVlLCJmaXJzdE5hbWUiOiJNaWNoZWxsZSIiImV4cGVyaWVuY2UiOiJCRUdJTk5FUiIsInVybF9pbWFnZSI6bnVsbH0.2wmIyWM024fj6PFDXKt5he6CFkQAqIF9UH4ebGUgEdY"}

Ensuite, je viens créer la requête de création de l'animal, je vais tout d'abord ajouter le JWT récupéré dans l'onglet "Authorization" de Postman.

ToutOPoils / Animals / Create an animal

POST http://localhost:3005/v1/animals

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Type Bearer Token Token eyJhbGciOiJIUzI1NiJ9.eyJpZCI6MjcsImFkbV...

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

J'ai sélectionné ici en type d'autorisation, le format "Bearer Token", de cette façon Postman va ajouter un header suivant ce format: "Authorization: Bearer [monJWT]", l'API sera alors capable de comprendre le format et de le traiter correctement.

Ensuite, je passe à l'ajout des données que je souhaite envoyer à mon API. Comme vu précédemment, pour ajouter une image je dois utiliser le type "formulaire", et non pas le type JSON comme pour le reste de mon API.

Dans Postman, je vais alors sélectionner le type "form-data", dans l'onglet "Body". Ensuite, j'y ajoute les champs que je souhaite passer en paramètre, ainsi qu'un champ de type "file", avec le nom "image".

**Send**

Params	Authorization	Headers (9)	<b>Body</b>	Pre-request Script	Tests	Settings	Cookies
<input type="radio"/> none	<input checked="" type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL		
KEY	VALUE						
<input checked="" type="checkbox"/> species	CAT						
<input checked="" type="checkbox"/> name	Stepan						
<input checked="" type="checkbox"/> bio	Taxe chat !						
<input checked="" type="checkbox"/> age	2021/03/13						
<input checked="" type="checkbox"/> gender	MALE						
<input checked="" type="checkbox"/> size	BIG						
<input checked="" type="checkbox"/> volunteer_experience	BEGINNER						
<input checked="" type="checkbox"/> box_id	1						
<input checked="" type="checkbox"/> image	stepan_the_cat.jpeg						
<input checked="" type="checkbox"/> tags	4,5,6						
Key	Value	Description					

Les données attendues lors de la création de l'animal via l'API doivent respecter les critères suivants:

- Réponse au format JSON
- Un [statut HTTP 201](#)
- Les données entrées
- L'identifiant de l'animal en base de données
- L'URL de l'image, pointant vers un serveur S3 d'AWS.

J'effectue ma requête et j'observe la réponse de mon API avec Postman:

ToutOPois / Animals / Create an animal

Save ...

**POST** http://localhost:3005/v1/animals **Send**

Params	Authorization	Headers (9)	<b>Body</b>	Pre-request Script	Tests	Settings	Cookies
<input type="radio"/> none	<input checked="" type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL		
KEY	VALUE						

Body	Cookies	Headers (8)	Test Results	Status: 201 Created Time: 400 ms Size: 549 B	<a href="#">Save Response</a>
Pretty	Raw	Preview	Visualize	JSON	
<pre> 1 2   "id": 58, 3   "species": "CAT", 4   "name": "Stepan", 5   "url_image": "<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/d9514201-70f0-4279-bf0f-d38a348f244f.jpeg">https://animalspictures.s3.eu-west-3.amazonaws.com/d9514201-70f0-4279-bf0f-d38a348f244f.jpeg</a>", 6   "bio": "Taxe chat !", 7   "gender": "MALE", 8   "size": "BIG", 9   "volunteer_experience": "BEGINNER", 10  "box_id": 1, 11  "age": "2021-03-12T23:00:00.000Z" 12 </pre>					

La réponse de mon API est un statut HTTP 201 - Created, au format JSON. Le format JSON contient bien les données entrées, ainsi que l'ID de l'animal et un lien vers notre stockage S3.

Pour vérifier que tout est correcte, je vais vérifier l'état de notre base de données, ainsi que sur l'application.

	<b>id</b> [PK] integer	<b>species</b> text	<b>name</b> text	<b>gender</b> text	<b>age</b> timestamp with time zone	<b>size</b> text	<b>volunteer_experience</b> text	<b>box_id</b> integer	<b>url_image</b> text
28	32	DOG	Chanel	FEMALE	2020-08-06 00:00:00+02	BIG	MEDIUM	8	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/61482/sierra-61482-6">https://www.la-spa.fr/app/assets-spa/uploads/animals/61482/sierra-61482-6</a>
29	34	DOG	Pluto	MALE	2020-01-31 13:24:53+01	BIG	BEGINNER	7	[null]
30	35	DOG	Elvira	FEMALE	2020-08-10 00:00:00+02	BIG	EXPERT	11	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/68558/cayenne-68558">https://www.la-spa.fr/app/assets-spa/uploads/animals/68558/cayenne-68558</a>
31	36	DOG	Cactus	MALE	2021-01-19 00:00:00+01	MEDIUM	MEDIUM	8	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/20554/cactus-20554">https://www.la-spa.fr/app/assets-spa/uploads/animals/20554/cactus-20554</a>
32	37	DOG	Rox	MALE	2014-06-01 00:00:00+02	BIG	MEDIUM	9	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/27751/rost-lady-27751">https://www.la-spa.fr/app/assets-spa/uploads/animals/27751/rost-lady-27751</a>
33	38	DOG	Diego	MALE	2018-04-04 00:00:00+02	MEDIUM	BEGINNER	9	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/59614/scoubidou-59614">https://www.la-spa.fr/app/assets-spa/uploads/animals/59614/scoubidou-59614</a>
34	39	DOG	Rita	FEMALE	2008-01-03 00:00:00+01	BIG	BEGINNER	10	[null]
35	40	DOG	Wendy	FEMALE	2009-09-10 12:24:53+02	MEDIUM	BEGINNER	11	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/bf82bace-066d-44bd-b1">https://animalspictures.s3.eu-west-3.amazonaws.com/bf82bace-066d-44bd-b1</a>
36	41	DOG	Whisky	MALE	2020-01-26 00:00:00+01	MEDIUM	BEGINNER	7	<a href="https://i.pinimg.com/736x/5a/5f/b3/5a5fb3571e8614faafffe9f8198c273ad.jpg">https://i.pinimg.com/736x/5a/5f/b3/5a5fb3571e8614faafffe9f8198c273ad.jpg</a>
37	42	DOG	Floup	MALE	2022-01-01 00:00:00+01	MEDIUM	BEGINNER	11	<a href="https://www.la-spa.fr/app/assets-spa/uploads/animals/50277/floup-50277-62">https://www.la-spa.fr/app/assets-spa/uploads/animals/50277/floup-50277-62</a>
38	44	DOG	Kimbo	MALE	2014-12-22 00:00:00+01	BIG	MEDIUM	13	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/7d01fe3-6087-4e72-a">https://animalspictures.s3.eu-west-3.amazonaws.com/7d01fe3-6087-4e72-a</a>
39	46	DOG	Milou	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	8	<a href="https://upload.wikimedia.org/wikipedia/commons/e/e7/Fox_Terrier_Alex.jpg">https://upload.wikimedia.org/wikipedia/commons/e/e7/Fox_Terrier_Alex.jpg</a>
40	47	CAT	Bill	MALE	2016-04-01 00:00:00+02	BIG	BEGINNER	15	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/385fb890-dbbc-4ac7-8">https://animalspictures.s3.eu-west-3.amazonaws.com/385fb890-dbbc-4ac7-8</a>
41	48	CAT	Kubo	MALE	2022-09-01 00:00:00+02	MEDIUM	BEGINNER	14	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/662c76c7-e178-4e98-8">https://animalspictures.s3.eu-west-3.amazonaws.com/662c76c7-e178-4e98-8</a>
42	49	CAT	Mika	MALE	2016-01-01 00:00:00+01	MEDIUM	MEDIUM	16	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/e647967e-66df-4aaa-8">https://animalspictures.s3.eu-west-3.amazonaws.com/e647967e-66df-4aaa-8</a>
43	50	CAT	Hobbo	MALE	2005-12-12 00:00:00+01	BIG	EXPERT	17	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/2b19f0f8-f544-4b15-94">https://animalspictures.s3.eu-west-3.amazonaws.com/2b19f0f8-f544-4b15-94</a>
44	51	CAT	Pistache	MALE	2014-04-15 00:00:00+02	SMALL	EXPERT	14	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/5182ef48-e871-4e56-b">https://animalspictures.s3.eu-west-3.amazonaws.com/5182ef48-e871-4e56-b</a>
45	52	CAT	O'titi	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	15	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/eab0cf60-be45-491a-b">https://animalspictures.s3.eu-west-3.amazonaws.com/eab0cf60-be45-491a-b</a>
46	53	CAT	bambou	FEMALE	2023-03-01 00:00:00+01	SMALL	BEGINNER	6	[null]
47	56	CAT	malo	MALE	2023-01-01 00:00:00+01	SMALL	BEGINNER	1	[null]
48	58	CAT	Stepan	MALE	2021-03-13 00:00:00+01	BIG	BEGINNER	1	<a href="https://animalspictures.s3.eu-west-3.amazonaws.com/d9514201-70f0-4279-b">https://animalspictures.s3.eu-west-3.amazonaws.com/d9514201-70f0-4279-b</a>

## LISTE DES ANIMAUX

VALIDER

REVOIR LA LISTE DES ANIMAUX

X

F

S

A

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

<div style="position: absolute; top: 10px; right: 10px; background-color: #337AB7; color

# FICHE DE STEPAN



2  
Ans

Gros

Mâle

Doux Calin Calme

box: C2

## BIOGRAPHIE

Taxe chat !

## DERNIÈRES VISITES DU BOX

le 07/02/2023

Rien à signaler, c'est cool !

**BONNE**

le 06/02/2023

**BONNE**

le 05/02/2023

Tout s'est bien passé

**BONNE**

[EN VOIR PLUS](#)

# VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

## I. Veille technologique

### A. Prisma

Aucun des membres de notre équipe n'ayant d'expérience avec Prisma, nous avons dû lors de la réflexion sur les technologies que nous souhaitions utiliser pour réaliser notre projet.

Initialement, nous pensions nous orienter vers du SQL pur, mais nous pensions éventuellement à utiliser Prisma. Nous avons donc commencé par faire des recherches sur les avantages et les inconvénients de ces deux hypothèses. Le sujet étant assez "dogmatique", nous avons effectué nos recherches de notre côté avant de débattre entre nous. J'ai notamment utilisé les liens suivants:

- <https://www.prisma.io/docs/concepts/overview/why-prisma>: table de comparaison de Prisma avec les autres ORM, réalisé par l'équipe de Prisma
- <https://stackoverflow.com/questions/72109628/orm-or-raw-sql-which-one-is-better>

Au vu du temps qui nous était donné et la complexité que pouvait entraîner l'utilisation de requêtes SQL pure, nous avons opté pour Prisma. L'autre intérêt de Prisma résidait dans son outil de versionnage de base de données.

### B. SDK S3 et Multer

Ayant la charge de réaliser la fonctionnalité d'envoi d'images, j'avais déjà entendu parler de Multer et du service de stockage S3, sans pour autant l'avoir utilisé.

J'ai donc cherché à voir comment ces solutions pouvaient s'articuler avec notre projet et la base de code déjà existante:

- <https://levelup.gitconnected.com/file-upload-express-mongodb-multer-s3-7fad4dfb3789>

Après avoir vu cet article, je me suis référé à la documentation du SDK d'AWS S3 ainsi qu'à la documentation de Multer pour réaliser la mise en œuvre de cette fonctionnalité.

- <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-node-examples.html>
- [https://docs.amazonaws.cn/en\\_us/sdk-for-javascript/v2/developer-guide/getting-started-nodejs.html](https://docs.amazonaws.cn/en_us/sdk-for-javascript/v2/developer-guide/getting-started-nodejs.html)
- <https://www.npmjs.com/package/multer>

Initialement je pensais pouvoir transférer directement les données de l'image de la requête vers les serveurs S3 sans avoir à stocker les images sur le serveur de l'API ou en mémoire.

- <https://www.npmjs.com/package/multer-s3>

Cependant je me suis rendu compte lors de la mise en place de cette fonction que l'upload était effectué au niveau du middleware de Multer, or je souhaitais pouvoir vérifier au niveau du controller que l'utilisateur avait l'autorisation de téléverser une image, par exemple pour la création d'un animal, qui est réservé aux administrateurs.

J'ai donc décidé de faire machine arrière pour revenir à une solution plus classique, à savoir conserver l'image en mémoire, passer la requête au controller et si l'utilisateur a les droits, uploader l'image sur les serveurs S3 via son SDK.

### C. Nodemailer et serveur SMTP

L'une des fonctionnalités de notre application consiste à envoyer un mail à un utilisateur pour l'informer de son inscription sur l'application.

Je me suis tout d'abord penché sur la solution d'envoi d'email Nodemailer avec sa documentation, ainsi que d'autre ressources:

- <https://nodemailer.com/>
- <https://mailtrap.io/blog/sending-emails-with-nodemailer/>

Enfin, je souhaitais éviter de "polluer" la base de code de l'application, en "découplant" le template du mail de la fonction d'envoi du mail de nodemailer.

J'ai trouvé la possibilité d'utiliser un moteur de templates ([handlebars](#)) avec un simple adaptateur pour Nodemailer:

- <https://medium.com/how-tos-for-coders/send-emails-from-nodejs-applications-using-nodemailer-mailgun-handlebars-the-opensource-way-bf5363604f54>
- <https://www.npmjs.com/package/nodemailer-express-handlebars>

## II. Veille de sécurité

### A. JWT et cryptage du mot de passe utilisateur

Pour réaliser le système de connexion de l'utilisateur, nous avons choisi d'utiliser les Json Web Tokens (JWT), nous nous posons également la question de la sécurité du stockage des mots de passe des utilisateurs en base de données.

Après avoir effectué quelques recherches sur les façons de stocker les mots de passe, nous avons décidé de les hasher en utilisant la librairie [bcrypt](#).

- <https://blog.logrocket.com/password-hashing-node-js-bcrypt/>
- [https://cheatsheetseries.owasp.org/cheatsheets/Password\\_Storage\\_Cheat\\_Sheet.html#hashing-vs-encryption](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#hashing-vs-encryption)

Puis nous avons effectué des recherches sur les bonnes pratiques à adopter en matière de sécurisation des JWT:

- <https://curity.io/resources/learn/jwt-best-practices/>
- [https://cheatsheetseries.owasp.org/cheatsheets/REST\\_Security\\_Cheat\\_Sheet.html#jwt](https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html#jwt)
- <https://stackoverflow.com/questions/27301557/if-you-can-decode-jwt-how-are-they-secure>

Enfin, nous avons cherché à savoir quelles sont les bonnes pratiques en matière de transmission du JWT entre le front et le back.

- <https://levelup.gitconnected.com/bearer-token-authentication-and-authorization-6e4d16890833>

## B. Utilisation des captchas et sécurisation des requêtes

Lors de la réalisation de la partie front-end, nous nous sommes posé la question de savoir si l'utilisation de captchas au niveau du formulaire de connexion était pertinente.

- <https://datadome.co/bot-management-protection/traditional-captcha-obsolete/>
- <https://www.cloudflare.com/learning/bots/how-captchas-work/>
- <https://support.google.com/a/answer/1217728?hl=en#:~:text=CAPTCHA%20helps%20protect%20you%20from,into%20a%20password%20protected%20account.>

Après ces recherches, nous avons compris que les captchas étaient une première protection au niveau de la partie front de l'application pour empêcher des robots malveillants de pouvoir opérer des attaques de type brute force.

Cependant, cette protection ne concerne que la partie front-end de notre application, la partie back restant ouverte à des attaques. A titre personnel, j'aurais souhaité que nous mettions en place un système de rate-limiting au niveau de notre API, ainsi que la mise en place de règles plus contraignantes au niveau des CORS. Au vu du temps qui nous était imparti pour réaliser ce projet, nous n'avons pas eu le temps de procéder à ces ajouts, mais ces ajouts pourraient être intégrés dans une mise à jour de notre application qui respecterait au maximum les bonnes pratiques de sécurité établies par la fondation OWASP.

- <https://www.cloudflare.com/learning/bots/what-is-rate-limiting/>
- <https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/>
- <https://cheatsheetseries.owasp.org/>

## III. Processus de recherche et traduction

### A. Processus

Pour effectuer mes recherches, j'adopte en grande partie le processus de recherche suivant:

1. Requête en anglais, plutôt qu'en français, sur google avec des termes simples
2. Sélection des sources par rapport au classement personnel, ainsi que la date de création / mise à jour de la source
3. Analyse critique des informations contenues dans les pages sélectionnées

En règle général je sélectionne les ressources dans l'ordre suivant:

1. La documentation de la technologie
2. Un acteur ou un groupe d'acteurs connus et identifiés dans le domaine de la recherche, par exemple la fondation OWASP dans le cadre d'une recherche sur la sécurité
3. Les sites ayant attrait à la programmation, exemple: medium, blog LogRocket, etc.
4. Éventuellement des questions sur Stackoverflow, mais je préfère éviter Stack Overflow quand il s'agit de questions liées au code en lui-même, cependant j'apprécie les différences de points de vue sur des questions "théoriques".

Pour illustrer cette pratique, je vais reprendre l'interrogation de mon équipe quant à l'utilisation de prisma.

Dans un premier temps, j'effectue une requête simple pour aborder ce questionnement:

The screenshot shows a Google search results page for the query "prisma vs raw sql". The results are as follows:

- Prisma ORM**  
https://www.prisma.io › docs › concepts › overview
- Why Prisma? Comparison with SQL query builders & ORMs**  
Raw SQL: Full control, low productivity. With raw SQL (e.g. using the native pg or mysql Node.js database drivers) you have full control over your database ...  
https://www.prisma.io › components › prisma-client
- Raw database access (Reference) - Prisma**  
Learn how you can send raw SQL and MongoDB queries to your database using the raw() methods from the Prisma Client API.
- Stack Overflow**  
https://stackoverflow.com › questions › orm-or-raw-sq...  
**prisma - Orm or RAW sql which one is better? - Stack Overflow**  
May 4, 2022 · 1 answer  
ORM and Raw SQL both have their own pros and cons. ... Prisma is a new kind of ORM that fundamentally differs from traditional ORMs and ...  
Is Prisma ORM stack good for new webapp (Nextjs + Nodejs) Mar 14, 2022  
How to see Prisma query values - Stack Overflow Sep 8, 2022  
How can I achieve to this result with Prisma? - Stack Overflow May 6, 2022  
Prisma batch upsert with raw SQL - Stack Overflow Feb 1, 2023

Ensuite j'observe les résultats, la première est la documentation de Prisma, je garde cependant à l'esprit que ce sujet est ouvert à interprétation je décide donc de consulter cette page de la documentation de Prisma.

Mais ce sujet étant source de débats, je regarde également les questions de Stack Overflow pour tenter d'avoir d'autres points de vue.

## B. Ressource en anglais

Dans le cadre de cette recherche, voici l'extrait que j'ai choisi de traduire. Il est issu de la documentation de Prisma:

<https://www.prisma.io/docs/concepts/overview/why-prisma>

### **Raw SQL: Full control, low productivity**

With raw SQL (e.g. using the native [pg](#) or [mysql](#) Node.js database drivers) you have full control over your database operations. However, productivity suffers as sending plain SQL strings to the database is cumbersome and comes with a lot of overhead (manual connection handling, repetitive boilerplate, ...).

Another major issue with this approach is that you don't get any type safety for your query results. Of course, you can type the results manually but this is a huge amount of work and requires major refactorings each time you change your database schema or queries to keep the typings in sync.

Furthermore, submitting SQL queries as plain strings means you don't get any autocompletion in your editors.

## C. Traduction effectuée

### **SQL Pur: contrôle total, productivité faible**

Avec le SQL pur (par exemple en utilisant les connecteurs NodeJS de base de données telle que [pg](#) ou [mysql](#)) vous avez le contrôle total des opérations effectuées sur votre base de données.

Cependant, le fait d'utiliser des requêtes SQL textuelles vient impacter votre productivité, vous devez effectuer des tâches lourdes et chronophages (gérer la connexion, code "boilerplate", ...)

Un autre des inconvénients de cette approche réside dans le fait que vous n'avez pas de typage de données pour les résultats de vos requêtes. Bien sûr, vous pourriez typer les résultats manuellement, mais cela demande un travail énorme et demande une refonte importante de votre code dès lors que vous modifiez le schéma de votre base de données ou les requêtes que vous effectuez.

De plus, gérer vos requêtes SQL de façon textuelle implique que vous n'aurez pas d'autocomplétion de votre code dans votre éditeur.

## CONCLUSION

En conclusion, j'aimerais avant tout remercier Angélique, Luis, Mathilde et Denise qui ont été mes collègues sur ce projet. L'émulation d'idées et de techniques liée à nos expériences passées à permis de construire rapidement et efficacement la méthodologie de travail qui allait nous accompagner durant ce mois de création du projet, et ce en répondant à une problématique déjà vécue par Angélique, notre product owner. Le fait d'avoir mis en place ces bonnes bases de gestion très rapidement nous a permis de réfléchir et de co-construire les réponses que nous souhaitions apporter à la problématique de la gestion des actions de bénévolat au sein des antennes locales de la SPA, ou plus globalement des refuges animaliers.

Bien que ce projet nous posait des "défis" techniques, car nous abordions des technologies que nous avions peu ou pas du tout utilisées précédemment, nous avons réussi à cumuler la recherche sur le fonctionnement de ces technologies, avec le développement de notre projet.

Initialement, j'appréhendais cette période de construction en groupe, à distance, sur une durée à la fois longue, mais courte au vu de l'ambition du projet. A l'issue de cette période, j'étais absolument convaincu de l'importance de cette expérience, et les bonnes pratiques qu'elle a pu m'apporter pour mon expérience professionnelle.

## ANNEXE

Les repos du projet sont accessibles à cette adresse: <https://github.com/toutopoils>

Le projet est accessible à l'adresse suivante: <http://toutopoils.herokuapp.com/>

RÔLE	EMAIL	MOT DE PASSE
Administrateur	admin@toutopoils.fr	administrateur
Bénévole	benevole@toutopoils.fr	benevoles

Le front et le back sont déployés sur deux instances [Heroku](#), tandis que la base de données est hébergée sur [Render](#).