

**INF3710 - Fichiers et Bases de données**

**TP5 - Applications et Bases de données**

Groupe 4

1847125 – Julien Legault

1846754 – Kevin Pastor

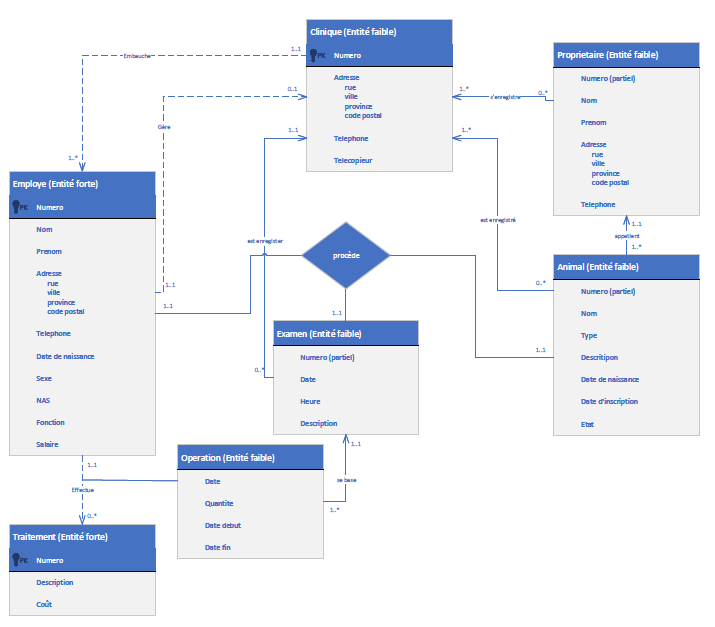
Soumis à : Manel Grichi

14 avril 2019

**1. Introduction**

Le but du laboratoire est de concevoir une base de données répondant aux critères d’une organisation offrant des soins pour animaux. La base de données doit être en mesure d’enregistrer plusieurs informations à propos des employées, des cliniques, des clients, des animaux, des traitements offerts, etc. De plus, une application web doit être développée afin de pouvoir manipuler les informations des animaux enregistrés.

**2. Modèle conceptuel**



*Figure 2.1 : Modèle conceptuel de la base de données*

**3. Modèle relationnel**

**3.1. Entité Employe**

Employe (numEmploye, nom, prenom, rue, ville, province, codePostal, telephone, dob, sexe, nas, fonction, salaire)

Primary Key (numEmploye)

L’entité *Employe* n’a pas d’association dirigé vers elle. C’est pour ça qu’elle ne possède pas de clé étrangère ou d’attributs provenant d’une autre entité.

**3.2. Entité Clinique**

Clinique (numClinique, numEmploye, rue, ville, province, codePostal, telephone, telecopieur)

Primary Key(numClinique)

Foreign Key (numEmploye) references Employe(numEmploye)

L’entité *Clinique* possède un clé étrangère représentant le numéro de l’employé gérant la clinique.

**3.3. Entité EnregistrementEmployeClinique**

EnregistrementEmployeClinique(numEmploye, numclinique)

Primary Key(numEmploye)

Foreign Key (numEmploye) references Employe(numEmploye)

Foreign Key(numClinique) references Clinique(numClinique)

L’entité *EnregistrementEmployeClinique* est nécessaire puisque les deux entités sont interdépendantes. Il faut alors créer une troisième table pour éviter une reférence cyclique.

**3.4. Entité Proprietaire**

Propriétaire (numProprietaire, nom, prenom, rue, ville, province, codePostal, telephone)

Primary Key(numProprietaire)

L’entité *Proprietaire* est une association binaire avec Clinique, alors il n’y a pas de clé étrangère.

**3.5. Entité EnregistrementProprioClinique**

EnregistrementProprioClinique (numProprietaire, numClinique)

Primary Key (numProprietaire, numClinique)

Foreign Key (numProprietaire) references Proprietaire(numProprietaire)

Foreign Key (numClinique) references Clinique(numClinique)

Cette entité doit être présente vue que nous avons une association binaire entre *Clinique* et *Proprietaire* nécessitant une nouvelle table.

**3.6. Entité Animal**

Animal (numAnimal, numProprietaire, nom, type, description, dob, doi, etat)

Primary Key(numAnimal, numProprietaire)

Foreign Key(numProprietaire) references Proprietaire(numProprietaire)

L’entité *Animal* necessite la cle du prorpitetaire pour etre unique et necessite alors une clé étrangère.

**3.7. Entité EnregistrementAnimalClinique**

EnregistrementAnimalClinique (numAnimal, numClinique)

Primary Key (numAnimal, numClinique)

Foreign Key (numAnimal) references Animal(numAnimal)

Foreign Key (numClinique) references Clinique(numClinique)

Cette entité doit être présente vue que nous avons une association binaire entre *Animal* et *Clinique* nécessitant une nouvelle table.

**3.8. Entité Examen**

Examen (numExamen, numClinique, numEmploye, numAnimal, date, heure, description)

Primary Key(numExamen, numClinique)

Foreign Key(numEmploye) references Employe(numEmploye)

Foreign Key(numAnimal) references Animal(numAnimal)

Foreign Key(numClinique) references Clinique(numClinique)

L’entité *Examen* a besoin des clés des entités *Employe*, *Animal* et *Clinique*, car elle dépend *Clinique* et elle est une association ternaire avec *Employe* et *Animal*.

**3.9. Entité Traitement**

Examen (numTraitement, description, cout)

Primary Key(numTraitement)

L’entité *Traitement* n’a pas d’association dirigé vers elle. C’est pour ça qu’elle ne possède pas de clé étrangère ou d’attributs provenant d’une autre entité.

**3.10. Entité Operation**

Opération (numTraitement, numExamen, numClinique, date, quantite, dateDebut, dateFin)

Primary Key(numTraitement, numExamen, numClinique)

Foreign Key(numTraitement) references Traitement(numTraitement)

Foreign Key(numExamen, numClinique) references Examen(numExamen, numClinique)

L’entité *Operation* a besoin des clés des entités *Traitement* et *Examen* car elle dépend de ces deux entités.

**4. Dépendances fonctionnelles**

**4.1. Entité Employe**

numEmploye → nom, prenom, rue, ville, province, codePostal, telephone, dob, sexe, nas, fonction, salaire

nas → nom, prenom, rue, ville, province, codePostal, telephone, dob, sexe, numEmploye, fonction, salaire

Dans l’entité *Employe*, la clé primaire *numEmploye* est unique faisant en sorte qu’aucune entrée de la table peut être dupliquée. Chaque colonne de la table représente un attribut unique et chaque entré pour chaque colonne est du même type. Cela fait en sorte que cette table est au moins 1NF. Puisque tous les attributs de l’entité dépendent de la clé primaire *numEmploye* et que la clé est composée d’un seul attribut, nous pouvons dire que la table est aussi 2NF. Puisqu’aucune dépendance transitive n’est présente dans l’entité, celle-ci est aussi 3NF. Par rapport au BCNF, le seul problème vient avec le déterminant *nas* qui n’est pas une clé candidate. Il faudrait enlever la clé primaire *numEmploye* et la remplacer par son *nas*. Comme ça le déterminant *nas* devient la clé candidate unique.

**4.2. Entité Clinique**

*numClinique → numEmploye, rue, ville, province, codePostal, telephone, telecopieur*

Dans l’entité *Clinique*, aucune entrée de la table ne peut être dupliqué à cause de la clé primaire *numClinique*. Aucun attribut n’est dupliqué et chaque entrée pour un même attribut représente une même information. L’entité est donc au moins 1NF. Chacun des attributs est étroitement relié à la clé primaire; l’entité est aussi 2NF. Puisqu’aucune dépendance transitive n’est présente dans l’entité, celle-ci est aussi 3NF. L’entité *Clinique* ne possède qu’un seul déterminant, *numClinique*, et celui-ci est une clé candidate faisant en sorte que l’entité est BCNF.

**4.3. Entité EnregistrementEmployeClinique**

*numEmploye → numClinique*

Pour l’entité *EnregistrementEmployeClinique*, la clé primaire *numEmploye* rend chaque entrée unique puisque celle-ci est unique dans l’entité *Employe*. L’entité est 1NF. Puisque l’entité ne possède qu’un autre attribut à part la clé primaire, l’entité est 2NF. Aucune dépendance transitive n’est présente dans l’entité, la rendant 3NF. Un peu pour la même raison que 2NF, l’entité est aussi BCNF.

**4.4. Entité Proprietaire**

*numProprietaire → nom, prenom, rue, ville, province, codePostal, telephone*

Pour l’entité *Proprietaire*, celle-ci est 3NF pour les mêmes raisons que l’entité *Employe*. Elle est cependant BCNF puisqu’elle ne possède pas d’attribut NAS ce qui implique que son seul déterminant, *numProprietaire*, est une clé candidate.

**4.5. Entité EnregistrementProprioClinique**

*numProprietaire → numClinique*

L’entité *EnregistrementProprioClinique* est BCNF pour les mêmes raisons que l’entité *EnregistrementEmployeClinique*.

**4.6. Entité Animal**

*numAnimal, numProprietaire → nom, type, description, dob, doi, etat*

L’entité *Animal* est BCNF pour les mêmes raisons que l’entité Propriétaire. Le déterminant *numAnimal* et *numProprietaire* est la clé candidate.

**4.7. Entité EnregistrementAnimalClinique**

*numAnimal → numClinique*

L’entité *EnregistrementAnimalClinique* est aussi BCNF pour les mêmes raisons que l’entité *EnregistrementEmployeClinique*.

**4.8. Entité Examen**

*numExamen, numClinique → numEmploye, numAnimal, date, heure, description*

L’entité *Examen* est BCNF pour les mêmes raisons que l’entité *Proprietaire*.

**4.9. Entité Traitement**

*numTraitement → description, cout*

*description → cout*

L’entité *Traitement* n’est pas 3NF puisque la description d’un traitement est unique à travers tous les traitements. Cela fait en sorte que la relation de transitivité *numTraitement* 🡪 *description* et *description* 🡪 *cout* peut être envelée en retirant l’attribut *numTraitement* de l’entité et en remplaçant la clé primaire de celle-ci par *description*.

**4.10. Entité Operation**

*numTraitement, numExamen, numClinique → date, quantite, dateDebut, dateFin*

L’entité *Operation* est BCNF puisqu’un seul déterminant est présent, avec les attributs *numTraitement*, *numExamen* et *numClinique*, qui est une clé candidate.

La base de données normalisée devient :

Employe (nas, nom, prenom, rue, ville, province, codePostal, telephone, dob, sexe, fonction, salaire)

Clinique (numClinique, numEmploye, rue, ville, province, codePostal, telephone, telecopieur)

EnregistrementEmployeClinique(numEmploye, numclinique)

Propriétaire (numProprietaire, nom, prenom, rue, ville, province, codePostal, telephone)

EnregistrementProprioClinique (numProprietaire, numClinique)

Animal (numAnimal, numProprietaire, nom, type, description, dob, doi, etat)

EnregistrementAnimalClinique (numAnimal, numClinique)

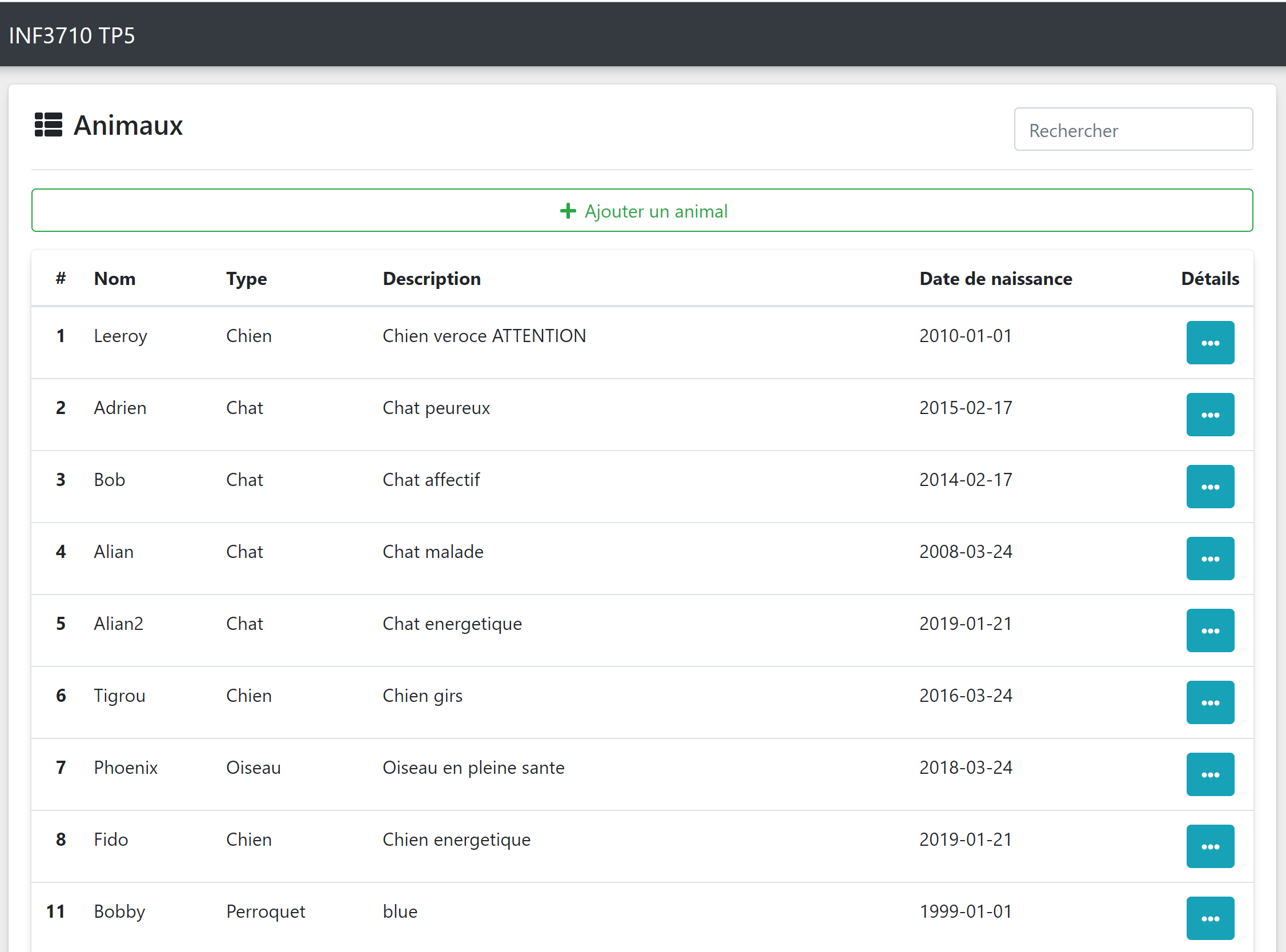
Examen (numExamen, numClinique, numEmploye, numAnimal, date, heure, description)

Examen (description, cout)

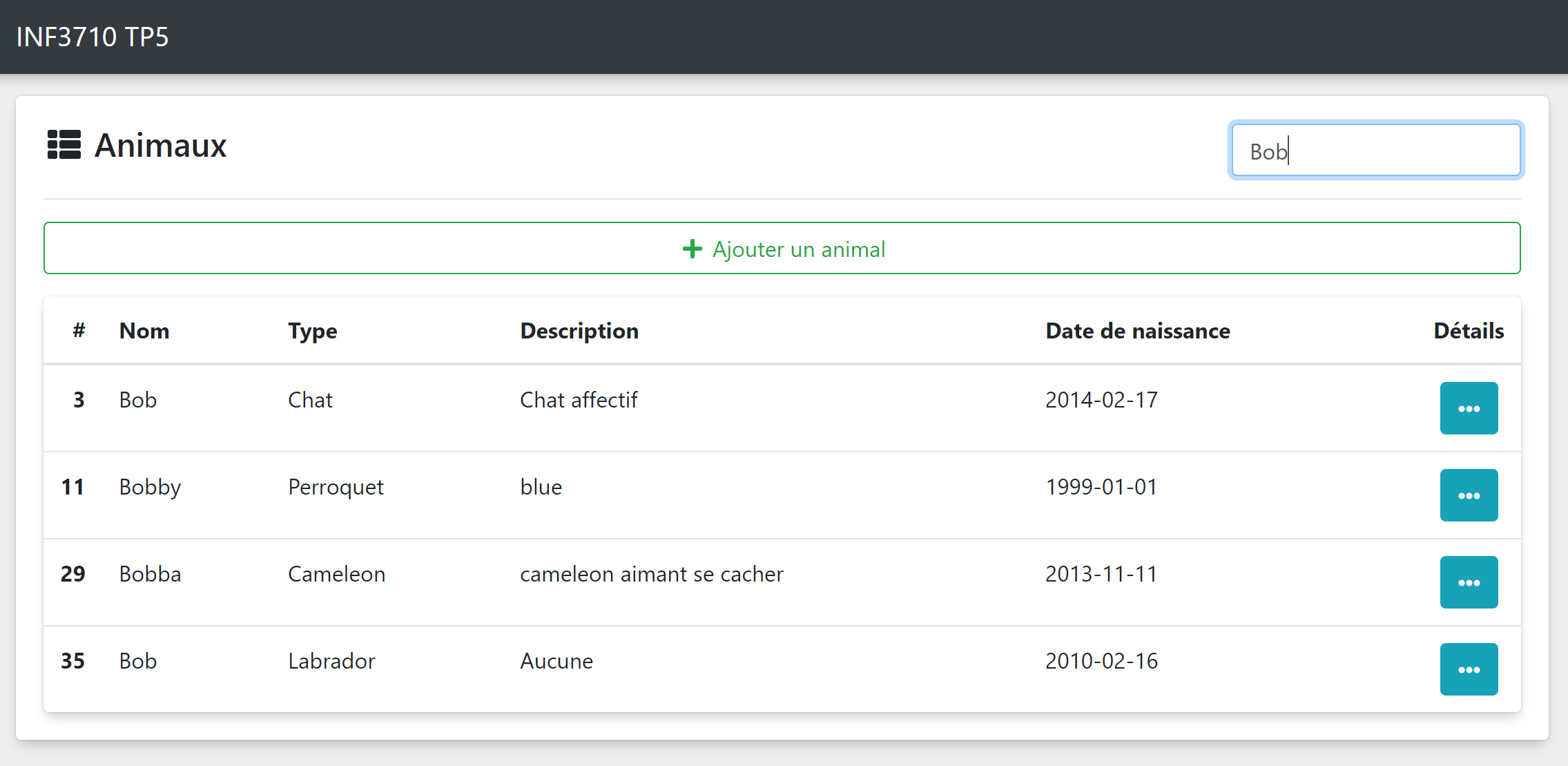
Opération (numTraitement, numExamen, numClinique, date, quantite, dateDebut, dateFin)

**5. Présentation de l’application**

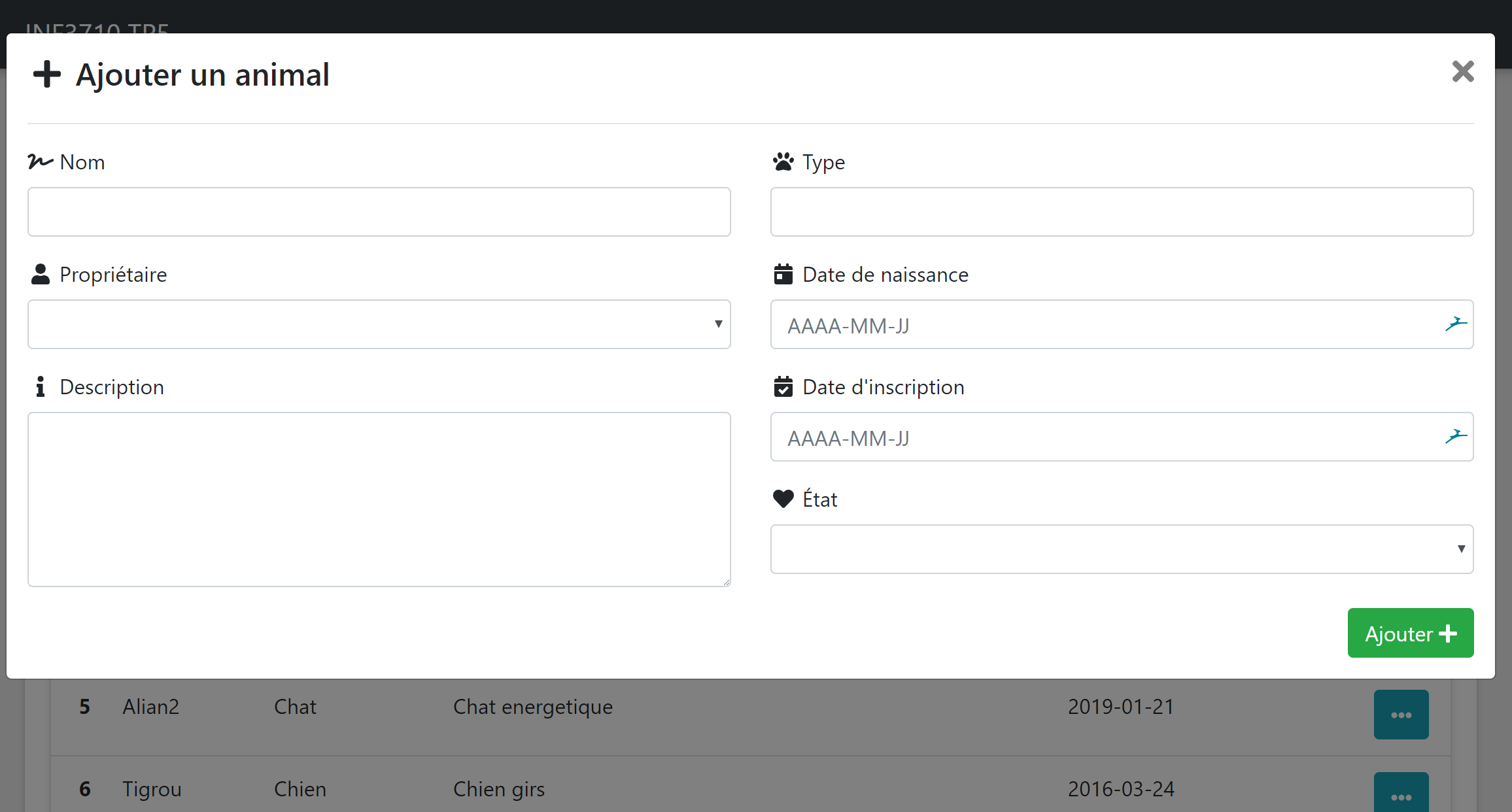
La page d’accueil présente la liste de tous les animaux enregistrés dans la base de données pour n’importe quelle clinique.



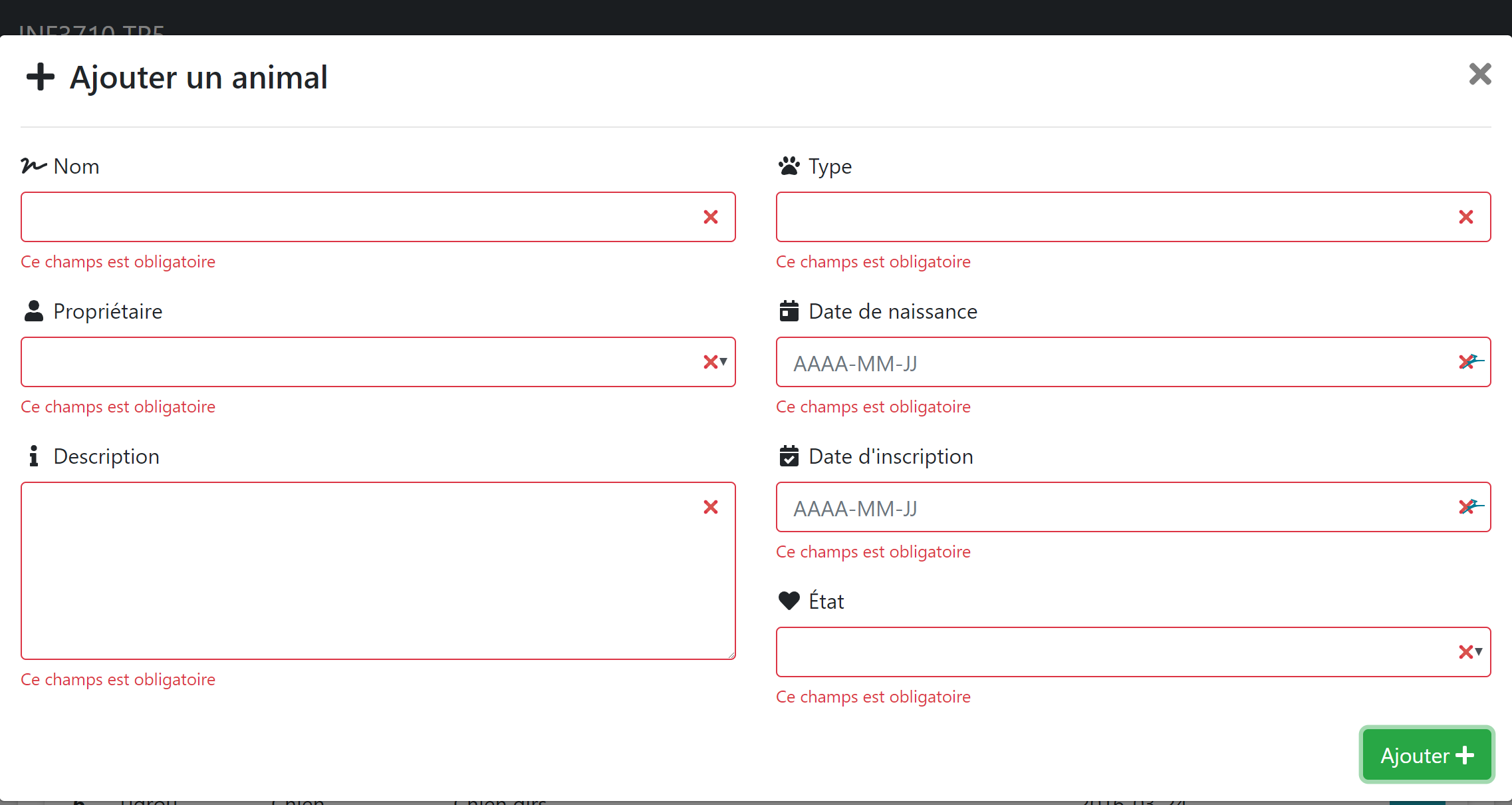
Un champ *Rechercher* situé en haut à droite de la liste d’animaux permet de filtrer la liste d’animaux et d’afficher seulement ceux dont le nom contient cette recherche. Ce champ de recherche est insensible à la case. Cette fonctionnalité permet à l’utilisateur de ne pas avoir à se soucier des majuscules ou des minuscules qu’un nom pourrait avoir.



Après avoir cliqué sur le bouton *Ajouter un animal* situé en haut de la liste d’animaux, une fenêtre est présentée avec tous les champs nécessaires à l’ajout d’un animal dans la base de données. Pour le champ *Propriétaire*, un menu déroulant est utilisé pour permettre à l’utilisateur de facilement sélectionner un propriétaire parmi la liste disponible.



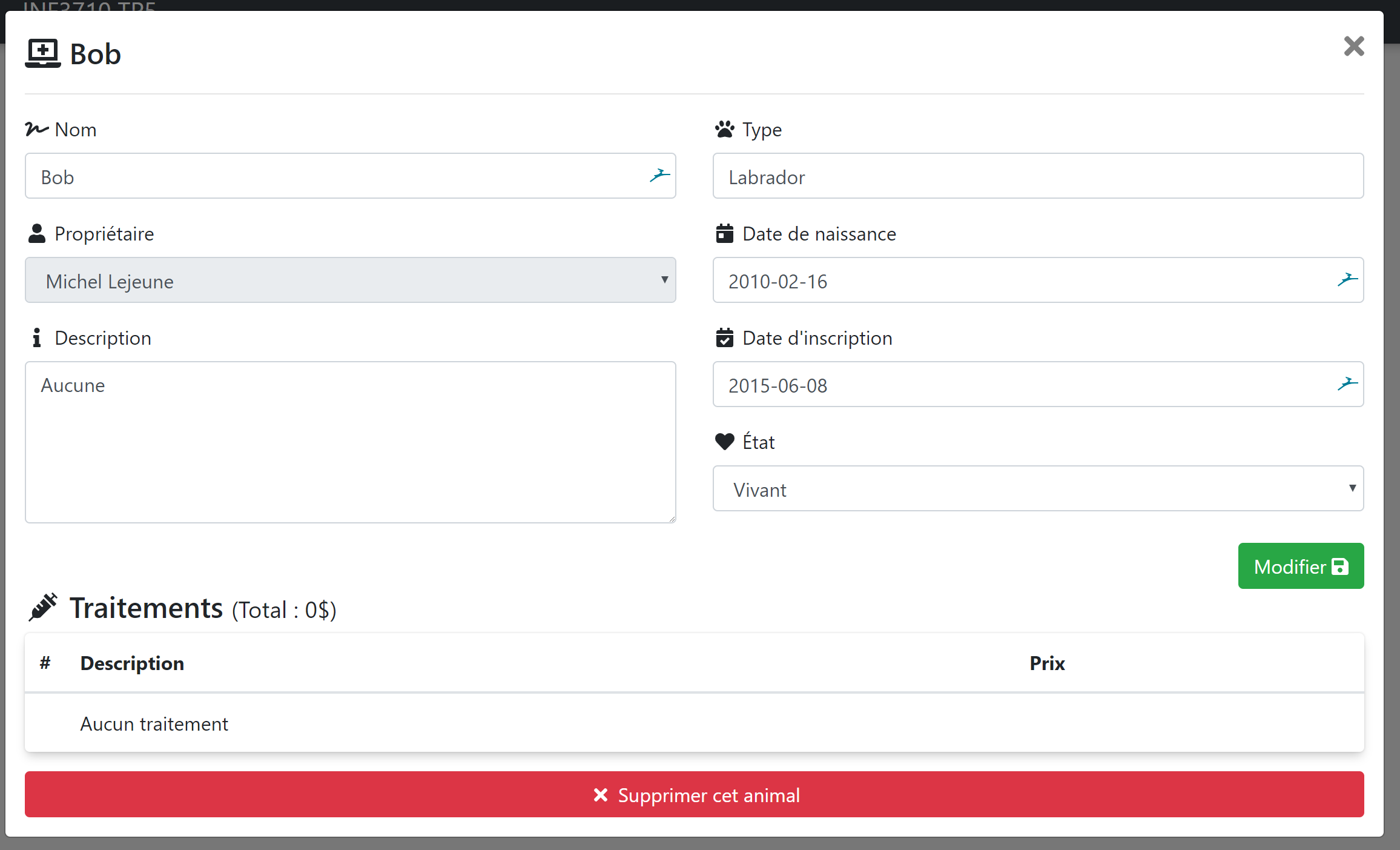
Lors de l’ajout d’un animal, si un champ n’est pas rempli ou bien qu’un champ ne respecte pas le format attendu, un petit message d’erreur est affiché à l’utilisateur pour lui permettre de corriger son erreur.



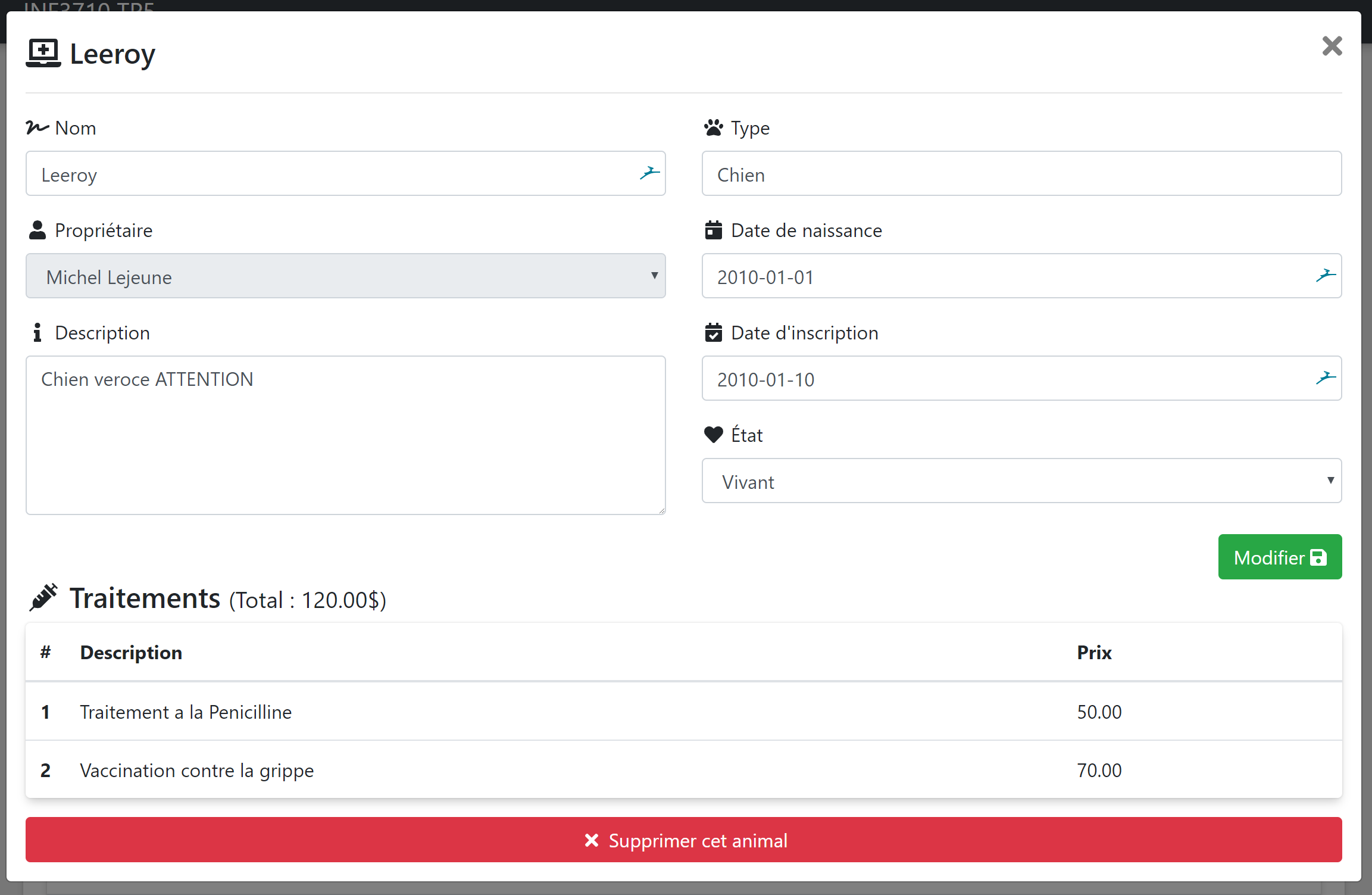
Après avoir cliqué sur le bouton *Ajouter*, la fenêtre d’ajout se ferme automatiquement et la liste d’animaux est mise à jour, faisant en sorte que l’utilisateur n’a pas besoin de rafraichir la page.



Sur n’importe quel animal présent dans la liste, l’utilisateur peut décider d’afficher plus de détails sur celui-ci en cliquant sur le bouton à sa droite. Une fenêtre s’ouvrira, affichant tous les détails disponibles sur cet animal. Si l’utilisateur souhaite modifier un de ceux-ci, il n’a qu’à le faire et ensuite cliquer sur le bouton *Modifier*. Puisque cette page de modification reprend la même structure que celle pour ajouter un animal, si l’utilisateur entre une mauvaise information, il en sera informé.



De plus, sur la page de détails, les traitements qui ont déjà été fait sur un animal, ainsi que la facture totale de ces traitements seront affiché en bas de la fenêtre. En regardant la capture d’écran précédente, on peut voir que si aucun traitement n’a été fait sur un animal donné, la liste affichera un message en conséquence. Si nous désirons supprimer un animal, il suffit de cliquer sur le bouton rouge *Supprimer cet animal*.



Nous avons mis un grand effort à présenter une interface simple, jolie et ergonomique pour rendre l’expérience utilisateur le plus intuitif possible. Nous nous sommes servis d’une plage de couleur traditionnel pour permettre à l’utilisateur de s’adapter rapidement à l’application. Les boutons modifiant des données sont présentés en vert, ceux en supprimant, en rouge, et les boutons d’actions sont présenté en bleu. Les icônes sont utilisées partout à travers l’application pour facilité la lecture de données ou d’informations sur la page. Les informations sont groupées de façon logique sans redondance inutile.

**6. Guide d’installation**

En ouvrant un terminal, il faut tout d’abord s’assurer d’être situé au bon emplacement, soit dans ./TP5/app.

Après s’être déplacé au bon endroit, il faut exécuter la commande npm install.

Après l’installation de toutes les dépendances du projet, il faut simplement exécuter la commande npm start et attendre que le client et le serveur se compile et se mettent en marche. Lorsque le client et le serveur seront opérationnel, une fenêtre de votre navigateur web par défaut devrait s’ouvrir et l’application devrait être opérationnelle.