Examen 5IDV2 - Juin 2022
Projet escale canine

Cugnon Geoffrey
Chargé de cours : **Delbar** Benjamin
IFOSUP Wavre 2021-2022

Objectif

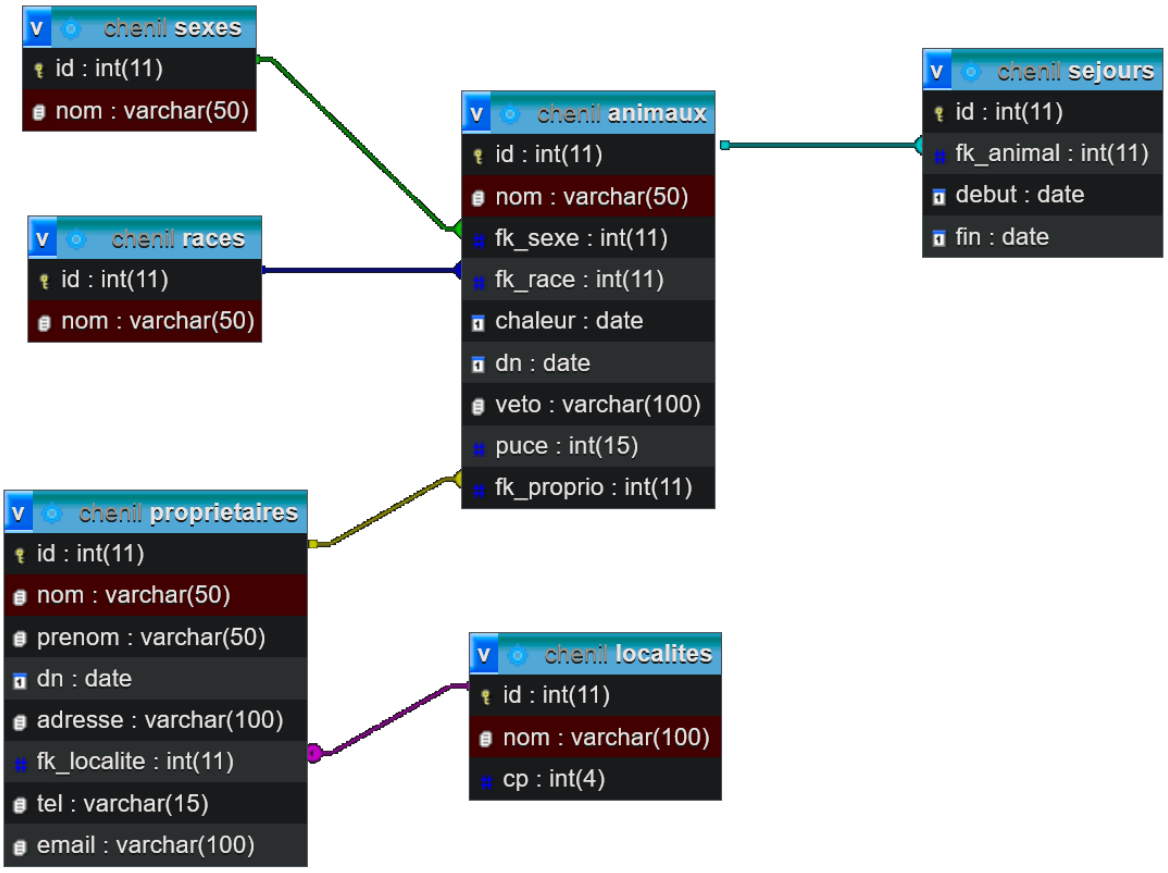
Il était demandé de réaliser une application permettant de gérer les séjours de nos amis canins dans un chenil. Cette application devait permettre de pouvoir :

- Consulter la liste des propriétaires ayant déposé leur animal en pension ainsi que leurs informations.
- Consulter la liste des animaux mis en pension ainsi que leurs informations.
- Consulter la liste des séjours au sein du chenil et leurs informations.

Cahier de charge

- Le projet devait fonctionner sur un serveur Apache avec une base de données mySQL et être écrit en PHP (version >= 7.4).
- Les tables de la base de données devaient respecter des principes de relation (restrict ou cascade selon).
- L'utilisation de framework était proscrite et la bibliothèque jQuery était obligatoire pour le javascript.
- Un Design Pattern (Stragey/Observer) devait être intégré au code.
- Le code devait respecter une architecture Modèle/Vues/Contrôleurs.
- Les formulaires devaient être sécurisés.
- Le projet devait comporter au moins une requête XHR.
- L'utilisation d'un routeur et d'un hôte virtuel était requise.
- Une navigation devait être présente sur les différentes pages.
- Les différentes relations doivent être en lazy load.

Schéma relationnel de la DB



Principales étapes

I. Réalisation de la base de données

J'ai tout d'abord commencer par réaliser la DB avec les tables reprises ci-dessus et je me suis assurer que les relations étaient correctement configurées selon les consignes :

- Restrict delete sur animal si présent dans séjour.
- Restrict delete sur propriétaire si a un animal.
- Cascade pour le reste des relations.

II. Création des entités dans le code

J'ai ensuite réalisé mes entités PHP en m'assurant qu'elles comportaient tous les attributs nécessaires pour recevoir les informations de la DB ainsi que leurs constructeurs et les fonctions relatives dont, entre autres, celles leur permettant de recevoir des informations des futurs DAO.

III. Création des DAO

Puis ce fut le tour des DAO faisant le lien entre les informations de la DB et les entités ainsi que des fonctions nécessaires à ceux-ci.

IV. Création des vues et des contrôleurs

J'ai alors commencé à créer mes vues en commençant par celle des propriétaires et j'ai ajouté au fur et à mesure les fonctions dont j'avais besoin pour gérer cette vue.
Je souhaitais que chaque vue (accessible par requête get) ait l'intégralité de son crud géré en XHR.
J'ai donc réalisé le javascript nécessaire pour y parvenir. Et j'ai ensuite répété ce processus pour chacune des vues selon le cahier de charge.

V. Création des vues et des contrôleurs

J'ai alors commencé à créer mes vues en commençant par celle des propriétaires et j'ai ajouté au fur et à mesure les fonctions dont j'avais besoin pour gérer cette vue.
Je souhaitais que chaque vue (accessible par requête get) ait l'intégralité de son crud géré en XHR.
J'ai donc réalisé le javascript nécessaire pour y parvenir… non sans mal. Et j'ai ensuite répété ce processus pour chacune des vues selon le cahier de charge.

VI. Création des vues et des contrôleurs

J'ai ensuite décidé de mettre en place le routeur… et quelle erreur de ne pas avoir commencé par ça. J'ai sous-estimé le travail que ça me demanderait d'adapter les requêtes XHR que j'avais alors décidé de pointer sur un api.php (là où les redirections get pointaient elles sur l'index.php). Mais bon, job's done.

Liste des routes

GET /proprietaires

Réponse	Type	Status	Description
Vue propriétaire + proprietaires/list_xhr.php	TEXT/HTML	200 OK	Renvoie la page complète avec la liste des propriétaires

POST /proprietaires/store

Réponse	Type	Status	Description
proprietaires/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue propriétaires avec la liste des propriétaires uniquement

POST /proprietaires/delete

Paramètre	Type	Status	Description
id	int	Obligatoire	Id correspondant à l'utilisateur à supprimer
Réponse	**Type**	**Status**	**Description**
proprietaires/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue propriétaires avec la liste des propriétaires uniquement

POST /proprietaires/update

Paramètre	Type	Status	Description
data + id	data(objet) + int	Obligatoire	Données concernant l'utilisateur à éditer + Id correspondant à cet utilisateur
Réponse	**Type**	**Status**	**Description**
proprietaires/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue propriétaires avec la liste des propriétaires uniquement

GET /animaux

Réponse	Type	Status	Description
Vue propriétaire + animaux/list_xhr.php	TEXT/HTML	200 OK	Renvoie la page complète avec la liste des animaux

POST /animaux/store

Réponse	Type	Status	Description
animaux/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue animaux avec la liste des animaux uniquement

POST /animaux/delete

Paramètre	Type	Status	Description
id	int	Obligatoire	Id correspondant à l'animal à supprimer
Réponse	**Type**	**Status**	**Description**
animaux/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue animaux avec la liste des animaux uniquement

POST /animaux/update

Paramètre	Type	Status	Description
data + id	data(objet) + int	Obligatoire	Données concernant l'animal à éditer + Id correspondant à cet animal
Réponse	**Type**	**Status**	**Description**
animaux/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue animaux avec la liste des animaux uniquement

GET /sejours

Réponse	Type	Status	Description
Vue sejours + sejours/list_xhr.php	TEXT/HTML	200 OK	Renvoie la page complète avec la liste des séjours

POST /sejours/store

Réponse	Type	Status	Description
sejours/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue séjours avec la liste des séjours uniquement

POST /sejours/delete

Paramètre	Type	Status	Description
id	int	Obligatoire	Id correspondant à un séjour à supprimer
Réponse	**Type**	**Status**	**Description**
sejours/list_xhr.php	TEXT/HTML	200 OK (done)	Met à jour un bout de la vue séjours avec la liste des séjours uniquement

Mise en avant d'une fonctionnalité

Je suis relativement content du résultat de mon affichage avec l'intégration du javascript. Ce n'est, certes, pas parfait, mais j'ai effectué un travail de recherche assez conséquent pour réussir à obtenir ce résultat en plus de ce qui nous avait été montré dans le cadre du cours.

J'aurais aimé n'avoir qu'une seule vue et gérer toutes mes pages en total javascript mais… un ami m'a indiqué que je ferais mieux de laisser ça de côté pour l'instant car j'aurais dû réaliser mon routeur en javascript également. Chose dont j'étais incapable dans le laps de temps imparti.

Autocritique

Il est vrai que je me suis contenté des fonctionnalités de base dans le cadre du cahier de charge, j'avais commencé à préparer mes fonctions where afin de commencer à travailler sur le filtrage de la liste des séjours selon le propriétaire… mais j'ai manqué de temps.

Dans l'idée, j'aurais dû ramener la fk_proprio de la table animal et m'en servir comme valeur à chercher dans la table propriétaire avec une fonction where et afficher nom,prenom du propriétaire dans la liste des séjours. Afficher un select avec les noms,prenoms des propriétaires des animaux en séjour et un bouton "Filtrer" selon le contenu. Évidemment, il aurait fallu réaliser ça en javascript pour avoir un affichage dynamique… Du boulot le JS !