420-15D-FX TP1 (30%)

Travail pratique #1 : NodeJS

Pondération : 30%

Travail à faire **individuellement**

Date de remise : **Jeudi le 24 mars avant minuit** (3 semaines)

Contexte

Ce travail pratique porte sur la complétion d'une API REST de base et la modification de la structure d'une base de données.

L'API devra respecter un jeu de tests automatisés et exhaustifs fourni par le professeur. L'API est partiellement complétée mais le développeur précédent n'a pas eu le temps de la terminer et de plus il a fait des erreurs !

Les technologies suivantes sont utilisées : Node.js, Express.js, MongoDB et Mongoose.

Description

Un refuge pour bébés chats souhaite développer une API afin de gérer les adoptions ainsi que les familles d'accueil temporaire. Une famille d'accueil temporaire s'occupe des chatons en attendant leur adoption.
La base de données comporte trois collections (**FamillesTemp**, **Adoptants** et **Chatons**).

L'API permet d'**ajouter**, de **modifier**, de **supprimer** et de **consulter** les familles d'accueil, les chatons et les adoptants (clients).

Un **Adoptants** peut :

- Adopter un chaton appartenant à une famille d'accueil.
- Un adoptant peut adopter plusieurs chatons.

Une **Famille d'accueil temporaire** peut :

- Accueillir des chatons.

Instructions

☐ Configurer l'environnement et les requêtes dans Postman avec les fichiers fournis :

TP2H2022.postman_collection.json

TP2H2022.postman_environment.json

☐ Charger les collections dans une base de données :

FamillesTemp.json

Chatons.json

Adoptants.json

☐ Compléter le code fourni dans le fichier projet.zip afin que tous les tests configurés dans Postman soient correctement passés.

☐ Compléter le code afin de respecter tous les éléments du fichier ressources en annexe.

Tests

Le jeu de test fourni utilise la version **initiale** de la base de données.

Si vous passez les tests après avoir modifier la base, il est normal que certains échouent.

Vous devrez recharger les données originales.

Caractéristiques générales du code

- Respectez les règles de base de la programmation vues en cours.
    - présentation du code
    - indentation
    - saut de ligne
    - nommage des fonctions et variables de manière intelligible et raisonné
    - etc.

- Une attention particulière sera portée aux commentaires descriptifs de votre application et à la qualité du code.
- Le code doit être le plus simple possible.

Procédure de remise

- Votre projet doit être remis sur Léa au format zip.
- Dans le dossier zip, vous devez ajouter une page de présentation du travail.

Annexe

Ressources REST

- Pour toutes les requêtes PUT et POST, les Header doivent avoir *Content-Type : application/json*

- Pour toutes les réponses, les Header doivent avoir *Content-Type : application/json* excepté pour la méthode DELETE.

- Le code de statut doit être 404 si une ressource n'est pas trouvée

URI	Méthodes	Description	Requête	Code de statut	Réponse
chatons	GET	**Affiche** tous les *chatons*. **Paramètre optionnel** : sexe type string permet de filtrer le chatons selon leur sexe		200	**Body** : Tableau d'objets Json *chaton*
chaton	POST	**Crée** un *chaton*.	**Body** : Objets Json *chaton*	201	**Body** : Objets Json *chaton*
chatons/:chatonId	GET	**Affiche** un *chaton* ayant l'id "chatonId".		200	**Body** : Objets Json *chaton*
chatons/:chatonId	PUT	**Modifie** un *chaton*	**Body** :	200	**Body** :

		ayant l'id "chatonId"	Objets Json *chaton*		Objets Json *chaton*
chatons/:chatonId	DELETE	**Supprime** un *chaton* ayant l'id "chatonId"		204	
famillesTemp	GET	**Affiche** tous les *chatons*. **Paramètre optionnel** : active type booléen permet de filtrer les familles qui accueillent présentement un ou des *chatons*		200	**Body** : Tableau d'objets Json *familleTemp*
familleTemp	POST	**Crée** une familleTemp.	**Body** : Objets Json *familleTemp*	201	**Body** : Objets Json *familleTemp*
famillesTemp/:familleTempId	GET	**Affiche** une familleTemp ayant l'id "familleTempId".		200	**Body** : Objets Json *familleTemp*
famillesTemp/:familleTempId	PUT	**Modifie** une familleTem ayant l'id "familleTempId"	**Body** : Objets Json *familleTemp*	200	**Body** : Objets Json *familleTemp*
famillesTemp/:familleTempId	DELETE	**Supprime** une familleTemayant l'id "familleTempId"		204	
adoptants	GET	**Affiche** tous les adoptants.		200	**Body** : Tableau d'objets Json *adoptant*
adoptant	POST	**Crée** un adoptant .	**Body** : Objets Json *adoptant*	201	**Body** : Objets Json *adoptant*
adoptants/:adoptantId	GET	**Affiche** un adoptant ayant l'id "adoptantId".		200	**Body** : Objets Json *adoptant*
adoptants/:adoptantId	PUT	**Modifie** un adoptant ayant l'id "adoptantId"	**Body** : Objets Json *adoptan*	200	**Body** : Objets Json *adoptant*

			t		
adoptants/:adoptantId	DELETE	**Supprime** un adoptant ayant l'id "adoptantId"		204	
adoptants/:adoptantId/adoption	PUT	Permet à un adoptant de réserver un chaton dans une famille d'accueil. Le chaton ne doit plus être associé à sa famille d'accueil suite à son adoption	Body : - id du chaton - date de l'adoption	200	**Body** : Objets Json *adoptant* contenant, dans son historiqueAdoption, l'objet représentant l'adoption (chatonId et dateAdoption)
famillesTemp/:familleTempId/accueillir	PUT	Permet à une famille d'accueillir un chaton.	Body : - id du chaton	200	**Body** : Objets Json *familleTemp*