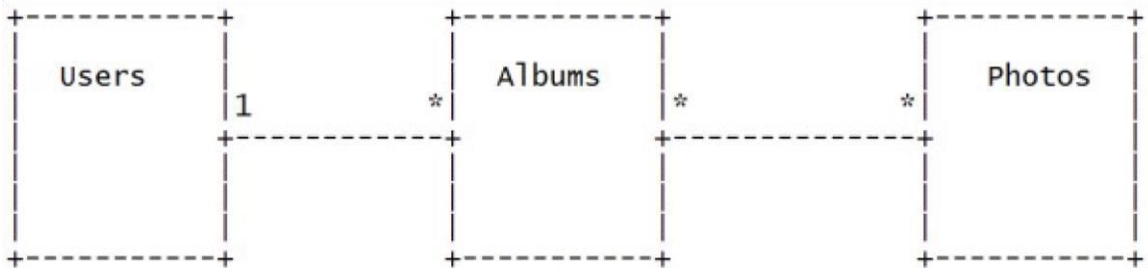
Afin de réaliser un site offrant un système de « galerie photos» à ses utilisateurs, il vous est demandé de réaliser une API REST(like) de gestion de « galerie » afin bien séparer l'application « frontend » du « backend ». Il vous est demandé d'utiliser le format d'échange de données JSON pour la communication des données vers les « clients ». Les données reçues par votre API seront envoyées par les futurs clients en JSON ou en paramètres (de GET, POST, ...) dans l'entête HTTP (la classe Input de Laravel gère déjà cela). Vous devez respecter le design pattern MVC au mieux durant votre développement.

1) Création des tables

Réalisez les « migrations Laravel » nécessaires à la création des tables de la base de données pour la gestion des galeries photos. Les tables de ces migrations devront respecter le diagramme de classe suivant :



Explication du diagramme de classe : les membres (users) peuvent créer plusieurs albums (relation 1-N). Un album est composé de plusieurs photos, mais les photos peuvent être dans plusieurs albums à la fois (relation « N-N » entre la table des « Albums » et celle des « Photos »). Les attributs minimums de ces tables sont :

- Pour les utilisateurs : un e-mail (identifiant) et un mot de passe
- Pour les albums : un titre, une description (texte) et des dates de création et de modification
- Pour les photos : le nom du fichier, une description (texte) et des dates de création et de modification. Avec en plus, bien sûr, les clés primaires et étrangères si nécessaires.

2) Models

Réalisez les « Models » (Data Table Gateway) nécessaires à l'interaction avec ces tables. Il est obligatoire d'utiliser l'ORM Eloquent de Laravel (y compris pour les jointures). Vous devez donc réaliser trois « Models » nommés User, Album et Photo et vous occupez des différents liens suivants : un utilisateur peut avoir plusieurs albums, un album appartient à un utilisateur, un album peut avoir plusieurs photos, une photos peut appartenir à plusieurs albums.

3) Seeding

Peuplez vos tables avec des données de test en utilisant le système de « seeding » de Laravel. Vous devez utiliser vos Models du point précédent. Ajoutez deux utilisateurs (avec leur mot de passe crypté), deux albums pour le premier utilisateur, un album pour le deuxième, trois photos pour le premier album du premier utilisateur, une de ces trois photos sera aussi présente dans le deuxième album du premier utilisateur. L'unique album du deuxième utilisateur contiendra 4 photos liées à aucun autre album.

4) Websocket pour la suppression d'une photo

La méthode de suppression d'une photo sera faite via WebSocket (ceci uniquement pour un but pédagogique). La première étape consiste donc à créer une nouvelle commande pour l'artisan Laravel (de la même manière que dans le TP sur les WebSocket) afin de lancer votre serveur WebSocket. Ensuite, et malgré le fait qu'il n'y ait qu'une seul action possible à votre WebSocket, réalisez une structure identique à celle du TP 5, c'est à dire en utilisant un dispatch « controller / action » pour rediriger vers une méthode d'un « controller ». Votre WebSocket **ne doit pas** gérer l'authentification. Tout client pourra donc supprimer une photo, même si elle ne lui appartient pas. L'unique action (de **PhotoController**) devra se nommer **destroy**. Elle sera appelée par les clients via un « send » respectant l'exemple JSON suivant :

{"action" : "*destroy*", "controller" : "*photo*", "id" : 1}

Votre méthode **destroy** devra faire les choses suivantes :

- Vérifier que la photo existe bien. Si ce n'est pas le cas, sortir de la méthode.

- Envoyer un message au client indiquant le OK pour la suppression.

- Supprimer la photo de la base de données.

- Faire un broadcast à tous les clients du JSON suivant:

{"action" : "*destroy*", "controller" : "*photo*", "id" : 1}