Travail pratique #3 : GraphQL

Pondération : 15 %

Travail devant être réalisé **en équipe de 2**

Remise : Au plus tard **dimanche le 18 mai** avant minuit

1 Objectif

Ce travail vise développer la couche GraphQL sécurisée avec Laravel Lighthouse sur l'API de films développé durant la session. En plus d'une remise du travail dans LEA, un dépôt Git pour l'équipe devra être maintenu de façon continue tout au long de celui-ci.

Ce travail nécessitera une bonification du modèle de données afin d'intégrer les données d'une source externe. Par la suite, il visera à développer quelques accès GraphQL aux données dans un contexte autorisé, ou non, selon la nature de la route.

2 Code de départ

Créez un nouveau projet Laravel, installez GraphQL Lighthouse et REST API tel que vu au cours. Copiez les dossiers de départ donnés, ils contiennent les migrations/*seeds* fonctionnels du modèle de données fourni en annexe et sont donnés avec cet énoncé ainsi que les modèles Laravel.

3 Bonification de la BD

On utilisera une source externe de données (un fichier JSON afin de faciliter la disponibilité des données – *data_source.json*) qui offre des évaluations de nos films provenant de 3 sources différentes, voici un extrait :


```
"id" : 1,
"title": "ACADEMY DINOSAUR",
"release_year": "2006",
"length": 86,
"description": "A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies",
"rating": "PG",
"special_features": "Deleted Scenes,Behind the Scenes",
"image": "",
"language_id": 1,
"ratings" :
[
    {
        "source" : "Internet movie database",
        "score" : 84.2,
        "votes" : 2
    },
    {
        "source" : "Rotten tomatoes",
        "score" : 78.5,
        "votes" : 4
    },
    {
        "source" : "Metacritic",
        "score" : 84.1,
        "votes" : 3
    }
]
```

Important : Dans ce fichier, les ids des films correspondent exactement aux ids de notre BD actuelle.

Puisque notre service web permet également d'évaluer ces films, nous allons maintenir une statistique de la moyenne actuelle (notre service + le service externe) de tous les films

Travail à effectuer :

Tâches	✓
Créer un dépôt sur GitHub ou GitLab**.** Mon utilisateur git est « filiat00 », m'inviter sur le dépôt.	
Ajouter une table dans la BD afin de pouvoir compiler la moyenne des revues de nos films. On choisira les champs adéquats pour pouvoir maintenir cette statistique.	
À l'aide de la source de données et du fichier exemple, créez un *seed* pour cette nouvelle table (notez que nous n'avons aucune critique en ce moment dans notre BD, ceci facilitera la compilation des données provenant de la source externe).	
Roulez les migrations et les *seeds*, assurez-vous que votre nouvelle table contient les données adéquates.	

4 Accès GraphQL publics

Cette partie vise à développer des accès GraphQL publics aux données, c'est-à-dire ne nécessitant aucune authentification/autorisation.

Important : Pour chaque accès développé, vous devez fournir un exemple d'appel dans le fichier « **appels_EFCS.txt** » donné avec cet énoncé (fichier qui sera à remettre).

1) Mettez en place les objets dans le schéma
2) Complétez le schéma GraphQL pour :

Accès	Retour	✓
Consultation d'un film en particulier	Titre du film, infos de ses critiques (id, score, comment)	
Consultation de tous les acteurs d'un certain film	Titre du film, Infos des acteurs (last name, first name, birthdate)	
Recherche de films selon les critères suivants : • Mot-clé (dans le titre ou la description) • Année de parution minimum (inclusive) • Longueur du film entre 2 valeurs précisées Une limite de 10 films doit être retournée :	id du film, titre, description, année de parution, longueur, infos du paginateur	

• Il est possible de préciser la page pour accéder à 10 autres films • Tous les critères sont optionnels et si aucun n'est fourni, on retourne les 10 premiers films https://lighthouse-php.com/master/api-reference/directives.html#where		

5 Accès GraphQL authentifiés

Cette partie vise à développer des accès GraphQL authentifiés aux données. Tel que vu au cours, on utilisera l'authentification Sanctum et l'utilisation de jetons. Les routes d'authentification API REST seront les seules routes qui seront appelées dans le cadre de ce travail afin d'enregistrer, d'authentifier et de déconnecter un utilisateur.

Avant de commencer : Incorporez votre contrôleur d'authentification du TP2 et ses routes, ajustez-le au besoin. Tel que vu au cours, configurez `lighthouse.php` afin qu'il supporte Sanctum.

Important : Pour chaque accès développé, vous devez fournir un exemple d'appel dans le fichier « **appels_EFCS.txt** » donné avec cet énoncé.

1) Élaborez le schéma GraphQL pour :

Accès	Retour	✓
Consulter les informations de l'utilisateur authentifié	Infos de l'utilisateur (id, email, first name, last name)	
Consulter les critiques de l'utilisateur authentifié	Infos de ses critiques (id, score, comment)	
Ajout d'une critique pour l'utilisateur authentifié. On devra mettre la nouvelle statistique pour le film en question. Ne pas oublier la validation. Indice : Il existe une directive Lighthouse pour injecter le id de l'utilisateur connecté dans le contexte de la mutation	Infos de la statistique (selon votre modèle)	
Seulement si admin : Ajout d'un acteur avec possibilité de : • Se lier à un ou plusieurs films • Mettre à jour l'image (seulement) de films	Infos de l'acteur (last name, first name, birthdate). Id et titre de ses films	

6 Critères d'évaluation

Éléments	Pondération
Types	8.5
Queries	4.5
Mutations	8
Inputs	6
Requêtes du fichier « appels_EFCS.txt »	3
Seeder de statistiques	2
Total	32

7 Documents à réaliser et à remettre

Veuillez remettre sur « LÉA » une archive « zip » contenant tous les fichiers du projet :

- Lien GitHub ou GitLab avec accès pour l'utilisateur filiat00
- Projet Laravel (sans les dépendances – sans le dossier *vendor*) dans LEA
- **Fichier « appels_EFCS.txt » inclus dans l'archive .zip**

Bon succès!

Annexe – Base de données (modèle de départ)