Examen 1 - formatif

Consignes

- Vous avez droit à toutes vos notes, exercices et Internet.
- Aucune communication n'est permise (messagerie, courriel, signaux de fumée, mimes, etc.). Les élèves pris en flagrant délit se verront attribuer la note de 0.
- La durée de l'examen est de 2 périodes (1h50)
- Une fois l'examen terminé, vous devez me remettre **tout le dossier de votre examen** via GitHub Classroom.
- Il est de votre devoir de vous assurer que vos fichiers sont inclus dans la remise.
- Aucun fichier ne sera accepté une fois que vous aurez quitté le lieu de l'examen.

Vous devez créer une API REST en Node.js avec la librairie Express.js et une base de données MySQL. La création de la base de données et l'insertion des données initiales est incluse dans un script SQL fournis dans le dépôt de l'examen. Votre api ne comportera qu'une route de sélection d'une liste de titre filtrée et paginée.

Installation

- Clonez le dépôt Github Classroom disponible à l'adresse que votre enseignant va vous donner.
- Exécutez le script SQL **dump_examen1_formatif.sql** présent dans le dépôt.

Configuration du projet

- Initialisez votre projet en important les librairies nécessaires et en créant une structure de fichier qui sépare le traitement en routes, contrôleurs et modèles.
- Sécurisez votre projet en vous assurant qu'aucunes données sensibles ni fichiers superflus ne sera téléversés sur GitHub.

Sélection de la liste des films ou séries

Dans la table *netflix_titles* de la base de données vous avez une série de titre disponible sur la plateforme Netflix. Chaque titre est catégorisé comme un film (« *Movie* ») ou une série (« *TV Show* ») par la valeur du champ « show_type ». Vous devez créer une route qui permettra de sortir une liste paginée de films ou de séries.

Ajoutez la route suivante à votre API :

- Méthode : GET
- Nom de la route : /api/titres/[:type_titre]
- Paramètre de la requête : page=1

Le paramètre de route *type_title* peut recevoir les valeurs *film* ou *serie*. Si une autre valeur est spécifiée retournez le message d'erreur « Le type [valeur du paramètre] est invalide » (voir plus bas)

Le paramètre de requête *page* permet de spécifier qu'elle page de résultat sera affichée. Chaque page comporte 10 titres. Si la valeur est invalide (plus petite que 1 ou qui n'est pas un nombre) retournez le message d'erreur « La page est invalide » (voir plus bas)

Structure de la réponse

```
{
    "titres" : [
        {
            "show_id" : 1,
            "title" : "Dick Johnson Is Dead"
        },
        {
            "show_id" : 7,
            "title" : "My Little Pony: A New Generation"
        }
    ],
    "filtre" : "film",
    "page" : 1,
    "url_page_suivante" : "/api/titres/film?page=2"
}
```

La valeur de la clé *url_page_suivante* est la route à entrer avec les bons paramètres pour se rendre à la prochaine page, à vous de créer la bonne chaîne de caractère. Dans le cas où on serait à la dernière page, affichez la valeur **null**.

Structure du message d'erreur

Code de statut 400

```
{
    "erreur" : "Message d'erreur"
}
```

Grille d'évaluation

Critère		Niveau de performance			
		Satisfaisant	**Acceptable**	**Inadéquat**	**Non réalisé**
Initialisation correcte du projet avec une structure routes, contrôleur, modèle.	5	Le projet est bien initialisé ET Le projet est bien structuré ET Les modules sont tous installés adéquatement - 5 -	Deux des trois critères sont réussis. -4-	Seulement un des trois critères est réussi. - 2 -	Aucun critère n'est réussi. - 0 -
Configuration appropriée du système de gestion de versions.	5	Tous les ajouts nécessaires à la configuration ont été faits. - 5 -	La configuration n'est pas sécuritaire au niveau des données personnelles OU Des éléments non nécessaires sont présents dans la gestion de versions. - 3 -		La configuration du système de gestion ne répond pas aux standards. - 0 -
Programmation correcte de la réception des données d'entrée	10	Les paramètres sont bien récupérés dans la section requête, dans la route et dans le corps de la requête. - 10 -	Les paramètres sont bien récupérés dans deux des trois méthodes. - 6 -	Les paramètres sont bien récupérés dans seulement une des trois méthodes - 4 -	Il y a des erreurs dans la récupération de chacune des trois méthodes - 0 -
Choix approprié des clauses, des opérateurs, des commandes ou des paramètres dans les requêtes à la base de données	10	Les requêtes à la base de données sont bien formulées et utilisent des paramètres préparés. - 10 -	Certaines requêtes à la base de données sont inexactes ou n'utilisent pas de paramètres préparés. - 6 -		Toutes les requêtes à la base de données sont inexactes et n'utilisent pas de paramètres préparés. - 0 -
Manipulation correcte des données de la base de données	10	L'usager est créé avec les bonnes informations et permissions. - 10 -	Il y a 2 erreurs ou moins dans les informations de l'usager ou dans ses permissions. -6-	Il y a entre 3 et 4 erreurs dans les informations de l'usager ou dans ses permissions. - 4 -	Il y a 4 erreurs ou plus dans les informations de l'usager ou dans ses permissions. - 0 -
Programmation correcte de l'envoi des données de sortie	10	Toutes les routes retournent le bon code de statut et les données retournées sont au bon format. - 10 -	Toutes les routes retournent le bon code de statut mais certaines données retournées ne sont pas au bon format. - 6 -	Certaines routes retournent le bon code de statut ET certaines données retournées ne sont pas au bon format. - 4 -	Aucunes routes ne retournent le bon code de statut et les données retournées ne sont pas au bon format. - 0 -