CÉGEP SHAWINIGAN
Du savoir et des gens

TP3

420-W0V-SW - Développement d'applications Web transactionnelles

Pondération de toutes les parties : 17.5%

Travail à effectuer

Un magicien est entré en contact avec votre enseignant pour l'aider à utiliser une faille dans la dimension arcanique. Il prétend qu'une application web Restful API est l'outil dont il a besoin pour prendre le contrôle de cette faille. Cet outil permettra d'améliorer l'accès à la magie dans son monde. Nous allons utiliser tout ce que nous avons appris dans le cours pour essayer de l'aider.

Attentes

Pour ce travail, vous devez respecter les principes de l'architecture MVC. Puisque cette application sera remise au magicien à la fin du contrat(date de remise), une attention particulière devra être accordée à la qualité du code que vous allez produire.

Le magicien a spécifié que la meilleure façon pour créer des objets dans son monde était de créer des documents dans une base de données MongoDB. Votre application doit respecter cette condition.

Il est impératif d'implémenter une solution qui facilite la traduction des éléments lisible par les utilisateurs de l'application. Nous ne pouvons garantir si l'utilisateur parle anglais, français ou l'elfique. Il faut donc garder ce principe en tête lorsque vous allez utiliser des champs String.

Partie 1 : Programme de base

Classe Magicien

nom	type	description
id	String	identifiant du magicien. Il fait référence au _id de la base de données.
name	String	Le nom du magicien
appearance	Object	Objet javascript permettant à notre magicien de décrire son apparence. ex {hair: "long", hairColor: "green", robe: …}
stats	Object	Objet javascript permettant à notre magicien d'avoir des caractéristiques ex: {strength:10, …}
level	Number	Le niveau du magicien
schools	Array of String	Liste des écoles de magie dont le magicien est capable de lancer un sort. ex: ["illusion", "divination", …]
alignment	String	Alignement du magicien. ex: "Chaotique bon" ou "Chaotic good"
spellbooks	Array of Grimoire	Liste des grimoires que le magicien peut utiliser.

Classe Grimoire

nom	type	description
id	String	identifiant du grimoire. Il fait référence au _id de la base de données.
name	String	Le nom du grimoire
schools	Array of String	Liste des écoles de magie disponible dans le grimoire
spells	Array of Spell	Liste des sorts du grimoire.
owner	Magicien	Propriétaire et créateur du grimoire. Peut être null.

Classe Sort

nom	type	description
id	String	identifiant du Sort. Il fait référence au _id de la base de données.
name	String	Le nom du sort
level	Number	Niveau du sort
school	String	École de magie du sort
effects	Array of Effect	Liste des effets du sort.

Classe Effet

nom	type	description
id	String	identifiant du Sort. Il fait référence au _id de la base de données.
description	String	Description de l'effet du sort.
school	String	École de magie de l'effet.

types	Array of String	Un effet peut avoir plusieurs types : “bon”, “mauvais”, “chaotique”, etc. Dépend de l’effet. Ex : Un effet qui fait perdre les cheveux du lanceur du sort peut être considéré “mauvais” et/ou “chaotique”. Un effet qui redonne des points de vie est considéré comme “bon”.

Actions

Un Magicien peut:	
	Créer un sort (Aléatoire et/ou fait à la main)
	Créer un grimoire (Aléatoire et/ou fait à la main)
	Ajouter un sort à un grimoire
	Acquérir un grimoire existant
	Lancer un sort d’un de ses grimoires

Créer un sort

Le sort peut être généré de façon aléatoire. Si un élément est spécifié par le magicien, alors cet élément ne sera pas généré de façon aléatoire. Lorsqu’un magicien crée un sort, le niveau du sort ne peut pas être plus élevé que le niveau du magicien. L’école du sort doit se trouver dans les écoles du magicien. La quantité d’effets dépend du niveau du sort. Les effets secondaires d’un sort ne sont pas obligés d’être de la même école que le sort.

ex : {name: “Create water”, school: “abjuration”, level: 1, effects: [{d: “Create water”, s: “abjuration”, a: “create_water”}]}
Le sort créé n’aura aucun élément généré aléatoirement.

ex2: {name: “RandyRando spellosorum”}
le sort créé possède un niveau inférieur ou égal au magicien, une école parmi celles du magicien et un ou plusieurs effets aléatoires.

Créer un grimoire

Le grimoire peut être généré de façon aléatoire. Si un élément est spécifié par le magicien, alors cet élément ne sera pas généré de façon aléatoire. Lorsqu’un magicien crée un grimoire, les écoles de magie sont prises parmi les écoles de magie du magicien. La quantité de sorts du grimoire devrait dépendre du niveau du magicien.

Ajouter un sort à un grimoire

Un magicien peut ajouter un sort à un grimoire dont il est le propriétaire. Si l’école du sort ne figure pas parmi la liste des écoles du grimoire, la liste des écoles de magie du grimoire devra être mise à jour.

Acquérir un grimoire existant

Un magicien peut ajouter le grimoire d'un autre magicien dans sa liste de grimoires.

Lancer un sort d'un de ses grimoires

Un magicien peut lancer un sort qui se trouve dans l'un de ses grimoires. Pour lancer le sort, le sort doit appartenir à l'une des écoles de magie du magicien. Le magicien doit avoir un niveau égal ou supérieur au sort pour le lancer.

Logger

Pour la première partie du travail, vous devez ajouter un logger dans votre application. Vous devrez utiliser ce logger sur chacune des actions qu'un magicien peut poser.

Vous devez utiliser ce logger quand vous attrapez une erreur.

Partie 2

Dans la deuxième partie du travail, vous devez implémenter l'authentification d'utilisateur dans notre application. Pour ce faire, nous avons besoin d'enregistrer un utilisateur et un mot de passe.

User

nom	type	description
id	String	identifiant du Sort. Il fait référence au _id de la base de données.
username	String	nom de l'utilisateur
password	String	Pour des fins pratiques, nous allons l'enregistrer comme une string non chiffré.
Rôle	String	// admin, mage

Login

Vous devez ajouter une route pour permettre à un utilisateur de se connecter. Cette route devrait vous retourner un token de connexion que vous pouvez utiliser dans les headers de requête pour exécuter la requête en tant que cet utilisateur.

Magicien

Nous allons ajouter un champ userId à notre magicien. Lors de la création d'un magicien, nous allons prendre le id de l'utilisateur authentifié.

Modification

Maintenant, les grimoires peuvent seulement être modifiés par le user qui a créé le magicien. Un utilisateur administrateur peut effectuer des actions sur les grimoires même s'ils ne sont pas leur propriétaire.

Notre middleware d'authentification devrait être exécuter sur les routes pour créer des magiciens et pour modifier les grimoires. Les autres routes peuvent être appelées de façon anonyme.

I18N

Vous devez intégrer la librairie i18n dans l'application. Votre application doit supporter le français et l'anglais au minimum. Il est attendu que les éléments suivants soient inclus dans la traduction :

- Les messages envoyés au client
- Les descriptions des effets des sorts
- Les écoles de magie
- Les alignements des magiciens

Remise

La remise du travail se fera sur GitHub. Vous devrez merger vos branches dans une branche principale et pousser un tag "REMISE". Je vais prendre la dernière occurrence du tag, le plus proche de la date et l'heure de remise, comme étant la remise finale.

La date limite de remise est le **13 juin à minuit.**

Si ca ne fonctionne pas, vous pouvez :

- zip votre projet
- Supprimer les dossiers node_modules
- me l'envoyer par mio

4	3	2	1	0
Les éléments demandés sont effectués de façon exemplaire.	Les éléments demandés sont respectés et le code est de bonne qualité.	Les éléments demandés sont respectés, mais la qualité du code est à améliorer	Les éléments demandés sont incomplets ou trop peu élaborés.	Les éléments demandés sont absents.
Correction négative				
Il y aura une pénalité de 1 pt par console.log non justifiée. Un console.log justifié est un console.log(error) dans une section catch d'un try/catch Une autre justification est de transmettre de l'information par rapport à notre serveur dans la console. Ex : console.log("server started on port X"), console.log("Inserted in BD_TAB_NAME : ID")				

Évaluation

X	Qualité du code	Note /4
ARCHITECTURE MVC		
	Logique métier se trouve dans les modèles	
	Les intéractions avec la BD se trouve dans les modèles	
	Les communications client-serveur sont dans les contrôleurs ou les routes.	
BASE DE DONNÉES		
	Liaison de données avec les ids	
GESTION DES ERREURS		
	Middleware d'erreur	
ACTIONS DU MAGICIEN		
	Toutes les actions sont implémentés	
LOGGER		
	Logger utilisé pour chacune des actions du magicien	
	Logger utilisé lorsqu'une erreur est capté	
AUTHENTIFICATION		

	Utilisation de JWT	
	Classe user (lien avec magicien, utilisation d'un user/password)	
	Route de login qui renvoi un token	
	Route de "Créer un magicien" requiert une authentification	
	Route de "modifier un grimoire" requiert une authentification	
	Un grimoire peut être modifié que par un admin ou son créateur	
I18N		
	Les traductions sont dans les fichiers de la langue spécifié	
	Les réponses dépendent de la langue spécifié dans le header de la requête	
	Les 4 éléments demandés sont traduites	
Console.log non justifié (utilisez le debugger)		
	TOTAL :	