Haute Ecole Léonard de Vinci — Session de janvier 2023

Examen de l'UE : Développement Web – Questions spéciales

Titulaire : Laurent Leleux, Raphaël Baroni
Bloc : 3BIN

Date et heure : 20/01/2023 à 13h30
Locaux : Ch43 A017, A019, B22
Durée de l'examen : 3h

Consignes : sur machine, à cours ouvert, internet ouvert, communication entre étudiants ou avec l'extérieur interdits, utilisation d'outils d'IA (chatGPT, Copilot...) interdite.

/!\ Lisez attentivement les consignes et l'introduction AVANT de démarrer l'examen /!\

Consignes et informations générales

Vous avez à votre disposition :

- Internet : vous pouvez donc **consulter** Moodle, les solutions des séances, des tutoriels en ligne, des forums... Cependant, **toute communication entre vous ou avec l'extérieur est formellement interdite**, sera considérée comme tricherie, et est donc passible de sanctions lourdes conformément au règlement des études.
- De même, **l'utilisation d'outils d'IA tels que ChatGPT, Copilot... est formellement interdite !**
- Vos notes au format papier
- Vos notes au format électronique (USB), mais l'usage de clefs USB n'est techniquement pas garantie sur les machines d'examen.
- Le boilerplate de l'examen, disponible sur EvalMoodle. Il contient les sources d'un projet de base qu'il vous faudra améliorer (dossiers **api** et **webapp**) ainsi qu'un fichier db.json qui contient les données qui doivent être présentes dans votre DB MongoDB, et un fichier Readme.

Comment démarrer :

- Veuillez dézipper le boilerplate sur votre machine, sur le bureau. Ne travaillez pas sur le lecteur réseau, ce sera beaucoup trop lent.
- Connectez-vous à votre compte MongoDB Atlas, ou créez-en un si nécessaire. Créez dans MongoDB Atlas les collections que vous voyez dans le fichier db.json, avec toutes les données associées. Pour cela, dans MongoDB Atlas, utilisez le bouton « insert document », ou utilisez toute autre technique que vous connaissez (Compass, Robo 3T, ligne de commande…). Si nécessaire, **n'oubliez pas de remplacer les ID's de documents associés (FK's) par les bons ID's de votre DB.** Il faut donc insérer les documents dans le bon ordre, ou modifier les documents après insertion. Utilisez le type « ObjectId ». Si vous n'y arrivez pas, utilisez le type String, mais vous devrez alors aussi utiliser ce type dans Mongoose.

- Modifiez les variables d'environnement du projet « api » pour que vous puissiez vous connecter à votre db.
- Vous pouvez à présent travailler dans les deux dossiers du projet (api et webapp).
- Une fois l'examen terminé, faites une archive .zip de votre dossier « projet » **sans les dossiers « node_modules »**, et appelez-là « **web3_NOM_PRENOM.zip** ». Ensuite veuillez soumettre cette archive sur evalMoodle avant la fin de l'examen. Si vous avez fait les bonus, vous aurez deux projets (2 api et 2 webapp) dans votre archive.
- A moins d'avoir ajouté des images, votre projet ne doit pas faire plus qu'un MB. La soumission n'autorise **pas d'envoi de plus de 10 MB**.

Nous allons exécuter votre code sur un logiciel de **détection de plagiat. Toute tentative de fraude, tout partage de code... sera sévèrement punie.**

Des questions « bonus » sont disponibles à la fin de l'examen. Celles-ci sont facultatives, mais les points ainsi obtenus permettront de compenser des éventuelles erreurs/lacunes dans les questions « de base ». Cependant, comme durant les séances, ces questions sont un peu plus complexes, ou sortent de la matière vue en cours.

Introduction

Dans ce projet, nous allons développer une application qui permettra à un magasin de jeux de société de faire le suivi de leurs stocks et des ventes.

Le magasin souhaiterait pouvoir consulter son catalogue, changer les valeurs des stocks pour chaque jeu du catalogue, mais également encoder des ventes, et consulter la liste des ventes.

Dans notre exemple simpliste, on ne peut vendre qu'un seul jeu par vente, mais la quantité peut être plus que 1 par vente.

1. Fonctionnalités back-end à développer

1.1 API – ressources et routes de base (4 points)

Un squelette de l'API est fourni, cependant celui-ci ne contient aucune ressource et aucune route pour l'instant.

Ajoutez une ressource `games` (jeux), et une ressource `sales` (ventes). Ces ressources doivent être récupérées dans la DB à l'aide des schémas et modèles Mongoose. Pour connaitre les schémas, utilisez le fichier `db.json` qui vous est fourni, et que vous avez déjà injecté dans votre DB.

La ressource `sale` contient un id de `game`. Il s'agit en quelque-sorte d'une FK. Pour la modéliser, vous pouvez utiliser le type `ObjectId` de Mongoose dans votre Schéma, mais également `String` si vous préférez.

Pour chaque ressource, il faut les routes de CRUD suivantes :

- Find all
- Find one (by id)
- Delete one (by id)
- Insert one

Lors de l'ajout d'une vente, le prix ne doit pas être donné, mais est calculé automatiquement sur base du prix du jeu. Mettez également à jour le stock du jeu associé. La date de la vente doit être transmise par le front-end, et sera par défaut la date actuelle.

N'hésitez pas à utiliser le dossier `requests` avec vos requêtes (fichiers *.http) pour faire vos tests. Pour rappel, il faut l'extension « humao.rest-client » pour utiliser ces fichiers avec VSCode. Vous pouvez également utiliser postman, insomnia...

1.2 API – validation (2 points)

Lors de l'insertion d'une vente (`sale`) ou d'un jeu (`game`), assurez-vous qu'il contient bien toutes les informations utiles (cfr. `db.json`).

Lors de l'insertion d'une vente, assurez-vous que le jeu associé à la vente existe bien dans la DB et que le stock est suffisant. On ne voudrait pas insérer une vente d'un jeu qui n'est pas dans le catalogue, ou qui n'est plus de stock... Assurez-vous également que la quantité est valide (positive, avec un minimum de 1).

Lors de l'insertion d'un jeu, assurez-vous que le stock est positif ou nul.

Veillez à employer un statut d'erreur http pertinent en cas d'erreur. Le message n'est pas important.

2. Fonctionnalités front-end à développer

Attention, depuis la fin du cours, une nouvelle version de Ant.design est sortie. Dans cet examen, nous ne l'avons pas utilisée pour ne pas changer par rapport à ce qui a été travaillé durant l'année.

2.1 Context (4 points)

Créez un `Context` qui va contenir :

- La liste de tous les `games` (jeux)
- La liste de tous les `sales` (ventes)

Les jeux et les ventes sont chargés depuis l'API automatiquement au chargement du `Context`.

Pour faire les requêtes à l'API, utilisez et complétez les services fournis : `gameApi` et `saleApi`.

2.2 Données d'une vente et d'un jeu (1 point)

Pour afficher une vente avec le nom du jeu qui a été vendu, nous avons besoin d'une méthode supplémentaire dans le `Context` : `getSaleWithGame(id)` qui renvoi la vente, avec dedans une clef `game` qui contient le jeu associé.

Voici ce qu'on pourrait recevoir :

```
{
    "buyer": "Camille",
    "date": "2021-01-02T10:17:35.457+00:00",
    "quantity": 1,
    "total": 25.5,
    "game" : {
        "name": "Pandemic",
        "price": 25.5,
        "stock": 5
    }
```

```
}
```

De même, pour afficher un jeu avec la liste de toutes les ventes associées, nous avons besoin d'une méthode supplémentaire dans le `Context` : `getGameWithSales(id)` qui renvoi le jeu, avec dedans une clef `sales` qui contient un tableau avec toutes les ventes associées.

Voici ce qu'on pourrait recevoir :

```
{
    "name": "Pandemic",
    "stock": 5,
    "price": 25.5,
    "sales" : [
        {
            "buyer": "Camille",
            "date": "2021-01-02T10:17:35.457+00:00",
            "quantity": 1,
            "total": 25.5,
        }
    ]
}
```

2.3 Catalogue des jeux (2 points)

Faites en sorte d'afficher une liste de tous les jeux, en utilisant les données du `Context`. Cette liste peut être rudimentaire, inutile d'y ajouter du style. Seul le nom et le stock doit y être affiché.

2.4 Routage (3 points)

Modifiez l'application pour qu'elle soit composée de trois routes distinctes :

- `/help` : la page d'aide avec le manuel de l'application
- `/games` : contient la liste de tous les jeux
- `/games/id` : contient une page d'un jeu (id en paramètre), avec la liste de ses ventes, ainsi que le formulaire d'ajout d'une vente.

Utilisez pour cela le module `react-router`. Vous pouvez choisir la version utilisée.

Veillez à garder une structure cohérente de l'application (découpe en composants).

Faites en sorte que la navbar apparaisse tout le temps, sur chaque page, et que les liens redirigent correctement sur chaque page. Pour voir la fiche d'un jeu, il faut cliquer sur le nom du jeu dans la liste. Attention, l'application doit rester une Single Page Application.

Dans cet exercice, vous ne devez pas encore rendre fonctionnel la page d'un jeu (avec la liste des ventes…), ou faire une page d'aide exhaustive… Concentrez-vous sur le routage, et n'affichez pour l'instant que des gros titres sur les pages.

2.5 CRUD des ventes (4 points)

Rendez la page d'un jeu, avec la liste des ventes, ainsi que le CRUD… fonctionnelle. On doit pouvoir :

- Voir les informations d'un jeu (stock et prix)

- Visualiser la liste des ventes du jeu, triés par ordre de date décroissante (les ventes les plus récentes au-dessus)
- Supprimer une vente à l'aide du bouton "supprimer" à côté de chaque vente
- Ajouter une vente à l'aide d'un input pour le nom du client, d'un input pour la quantité, et d'un bouton « ajouter ».

La date d'une vente est la date à laquelle la vente a été ajoutée.

Après l'ajout d'une vente, la liste doit automatiquement être mise à jour avec la nouvelle vente, et le formulaire doit être reset.

Après avoir supprimé une vente, la liste doit automatiquement être mise à jour.

3. Fonctionnalités bonus (2 points bonus)

Voici quelques fonctionnalités supplémentaires que vous pouvez développer. Les points obtenus, mais également les erreurs faites ne pourront toujours qu'améliorer votre note, et donc rattraper certaines erreurs commises dans l'examen. ***Vous ne perdrez en aucun cas des points en essayant les fonctionnalités bonus ci-dessous.***

Si vous faites des fonctionnalités bonus, faites-les dans un nouveau projet ! *Copiez vos solutions précédentes, et faites* ***un nouveau dossier avec "BONUS" dans le nom****, contenant votre API et votre WebApp, et modifiez celui-ci. Vous aurez donc 2 API et 2 WebApp, une contenant les bonus, et l'autre sans les bonus. De cette manière, nous pouvons corriger les bonus indépendamment de votre examen, et vous ne perdrez pas de points en cas d'erreurs dans les bonus !*

3.1 Suppression en cascade

Lors de la suppression d'un jeu (au niveau API car vous n'avez pas dû le faire côté webapp), les ventes associées ne sont pas supprimées. Remédiez à cela, en supprimant en cascade toutes les ventes associées au jeu supprimé.

3.2 Date de la vente

A côté de l'input pour l'ajout d'une vente, ajoutez un datepicker Ant.design qui permettra de choisir la date et l'heure de la vente. Par défaut, si aucune date/heure n'est donnée, c'est la date/heure actuelle qui est utilisée.

Voici le lien dans la documentation pour faire un date-picker avec time :
https://ant.design/components/date-picker/#components-date-picker-demo-time

3.3 Page actuelle

Cette fonctionnalité est plus complexe !

Dans la NavBar, mettez en évidence le lien vers la page actuelle. Par exemple, si on est sur la page `/games`, le lien `Games` doit être mis en évidence.

Consultez la documentation du `react router` pour savoir quelle est le `path` de la page actuelle.

Consultez la documentation de Ant.Design pour voir comment réagit la navbar/menu.