Institut Paul Lambin Session de janvier 2022

Examen de l'UE : Développement Web – Questions spéciales

Titulaire : Laurent Leleux, Philippe Van Eerdenbrugghe
Bloc : 3BIN

Date et heure : 21/01/2022 à 13h45
Locaux : Ch43 A017, A019, A026, B22
Durée de l'examen : 3h

Consignes : sur machine, à cours ouvert, internet ouvert, communication entre étudiants ou avec l'extérieur interdits.

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

Consignes et informations générales

Vous avez à votre disposition :

- Internet : vous pourrez 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.
- 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 (Robo 3T, ligne de commande…). **N'oubliez pas de remplacer les ID's dans `event` (clef `child`) par les ID's du bon `child`.** Il faut donc insérer d'abord les enfants avant les événements. 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 aux crèches de faire le suivi des journées des bébés.

Dans les crèches, la plupart des événements qui concernent les bébés sont consignés dans des fardes, ou des cahiers qui sont ensuite transmis aux parents à la fin de chaque journée. Ceci permet aux parents d'être tenu informé de la journée de leur bébé. Nous allons informatiser ce processus.

Pour cela, nous allons avoir une liste d'enfants, et pour chaque enfant, une liste d'événements. Ces événements peuvent être encodés à l'aide d'un input. Par exemple : Changement de la couche, biberon, début de la sieste, fin de la sieste… Chaque événement a une date et heure précise.

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 `children` (enfants), et une ressource `events` (événements). 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 `event` contient un id de `child`. Il s'agit en quelque-sorte d'une FK. Pour la modéliser, vous pouvez utiliser le type « ObjectId » de Mongoose dans votre Schéma.

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

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

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'un événement (`event`) ou d'un enfant (`child`), assurez-vous qu'il contient bien toutes les informations utiles (cfr. `db.json`).

Lors de l'insertion d'un événement assurez-vous que l'enfant associé à l'événement existe bien dans la DB. On ne voudrait pas insérer un événement qui ne soit associé à personne…

Lors de l'insertion d'un enfant, assurez-vous que son `name` fasse au minimum 3 caractères.

Veillez à employer un statut d'erreur http pertinent en cas d'erreur.

2. Fonctionnalités front-end à développer

2.1 Context (4 points)

Créez un `Context` qui va contenir :

- La liste de tous les `children` (enfants)
- La liste de tous les `events` (événements)

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

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

2.2 Données d'un enfant (1 point)

Pour afficher le profil d'un enfant avec sa liste d'événements, nous avons besoin d'une méthode supplémentaire dans le `Context` : `getChildWithEvents(id)` qui renvoi l'enfant, avec dedans une clef `events` qui contient un tableau avec tous les événements de cet enfant.

Voici ce qu'on pourrait recevoir :

```
{
    "name": "Camille",
    "birthDate": "2021-01-02T10:17:35.457+00:00",
    "gender": "F",
    "events" : [
        {
            "date": "2022-01-20T12:00:00.000+00:00",
            "name": "Changement de lange"
        }
    ]
}
```

2.3 Liste des enfants (2 points)

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

- `/children` : contient la liste de tous les enfants
- `/children/id` : contient une page d'un enfant (id en paramètre), avec la liste de ses événements, ainsi que le formulaire d'ajout d'événement.

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 enfant, il faut clique sur le nom de l'enfant 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 enfant (avec la liste des événements…), 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 événements (4 points)

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

- Voir les informations d'un enfant
- Visualiser la liste des événements de l'enfant, triés par ordre de date décroissante (les événements les plus récents au-dessus)
- Supprimer un événement à l'aide du bouton "supprimer" à côté de chaque événement
- Ajouter un événement à l'aide d'un input et d'un bouton « ajouter ».

La date d'un événement est la date à laquelle l'événement a été ajouté.

Après l'ajout d'un événement, la liste doit automatiquement être mise à jour avec le nouvel événement, et le formulaire doit être reset.

Après avoir supprimé un événement, la liste doit automatiquement être mise à jour.

3. Fonctionnalités bonus (2 points bonus)

Si vous avez terminé tous les points ci-dessus et qu'il vous reste du temps, 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 enfant (au niveau API car vous n'avez pas dû le faire côté webapp), les événements associés ne sont pas supprimées. Remédiez à cela, en supprimant en cascade tous les événements associés à l'enfant supprimé.

3.2 Date de l'événement

A côté de l'input pour l'ajout d'un événement, ajoutez un datepicker Ant.design qui permettra de choisir la date et l'heure de l'événement. Par défaut, si aucune date/heure 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 /children, le lien "Children" doit être mis en évidence. Le lien "Help" ne peut pas 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.