Examen de première session

Durée : 1 heure 30

Documents autorisés : La refcard du module

La base de données de cet examen contient une table `notes` représentée dans la figure 2.

cours	etudiant	note
info	Bruce	12
info	Roger	9
maths	John	17
maths	Emilia	13
maths	Jessica	20
anglais	Natasha	5

FIGURE 1 – La table `notes`.

Exercice 1 (1.5 points) — *requête SQL en PHP*

Dans le répertoire racine du backend, on possède un script PHP `mysqlConnect.php` déjà écrit qui crée une instance `$pdo` de PDO, et qui la connecte à la base de données. Cette variable est globale.
Écrivez une fonction PHP `getNotes` qui prend en argument un nom de cours et qui renvoie la liste (etudiant, note) des étudiants correspondant au cours. Par exemple, si l'argument est égal à « info », la fonction renverra :

```
[['etudiant' => 'Bruce', 'note' => 12], ['etudiant' => 'Roger', 'note' => 9]]
```

Si le cours n'existe pas ou bien si aucun étudiant ne le suit, la fonction renvoie une liste vide.

Exercice 2 (1.5 points) — *Backend en PHP*

Dans le répertoire racine du backend, on possède deux scripts PHP :

1. le script `helper.php` des TP, dans lequel on a supprimé la partie concernant l'authentification ;
2. un script `getNotes.php` qui contient la fonction `getNotes` de l'exercice 1.

En exploitant ces scripts, écrivez dans le répertoire racine du backend un nouveau script `notes.php` qui est appelé par un frontend via une méthode **POST**. Ce dernier transmet à votre script un couple (`cours`,valeur) encodé au format `FormData`, où « valeur » est le nom d'un des cours de la base de données (par exemple, le couple (`cours`,info)). Votre script réalise alors les opérations suivantes :

1. Il vérifie que le couple (`cours`,valeur) a bien été transmis via un **POST**. Si ce n'est pas le cas, le script envoie un message d'erreur au frontend.
2. Sinon, il envoie une requête au serveur MySQL pour récupérer la liste (etudiant, note) des étudiants correspondant au cours. Par exemple, pour le couple (`cours`,info), la liste correspond aux 2 premiers éléments de la figure 2.
3. Il envoie au frontend un message de type `PhpData` (cf. les TP) contenant la liste calculée en 2.

Ici, pour simplifier, aucune authentification n'est réalisée.

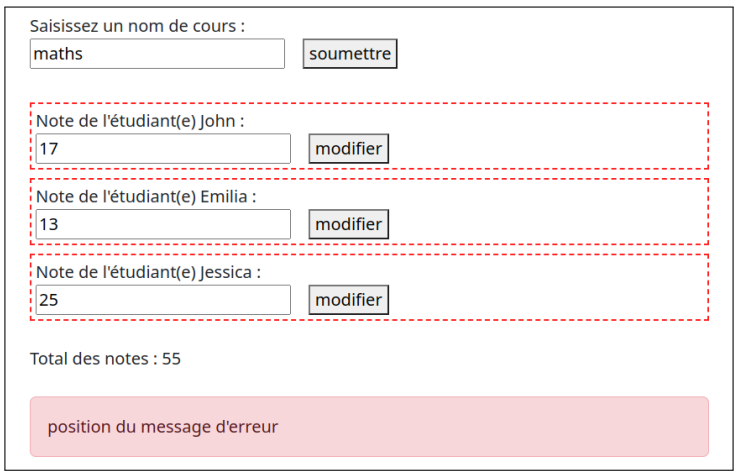

FIGURE 2 – Une page d'une application Angular

Exercice 3 (3 points) — *Service Angular*

Écrivez un service Angular `MessageService` contenant une méthode `sendMessage(url, data)` telle que `url` est le suffixe d'une URL et `data` est un objet javascript contenant des données sous la forme de couples (clé,valeur). La méthode envoie ces données à un backend PHP via un **POST** et renvoie son résultat. La véritable url du backend est la concaténation de `http://127.0.0.1:5000/back` et du suffixe `url`. Vous ferez en sorte que les variables de session du backend puissent être utilisées.

Exercice 4 (4 points) — *Composant Note en Angular*

Dans cet exercice, on souhaite réaliser le composant `Note` entouré par des pointillés en rouge de la figure 2. Ce composant récupère de son parent les données (étudiant,note) d'un étudiant. comme indiqué sur la figure, il contient un « `input` » contenant le note de l'étudiant(e) transmise par le parent. On peut modifier celle-ci. Dans ce cas, si l'on clique sur le bouton `modifier`, on doit calculer la différence entre la nouvelle valeur et l'ancienne et la transmettre via un événement `modif` au parent. Par exemple, si la note de Jessica est passé de 20 à 25, on transmet au parent la valeur $5 = 25 - 20$. Écrivez le contenu du template HTML ainsi que la classe TypeScript du composant `NoteComponent`. En ce qui concerne le .ts, vous n'indiquerez que le contenu de la classe. Il est donc inutile d'écrire les `import` ou le décorateur `@component`.

Exercice 5 (7 points) — *Page Angular*

On souhaite réaliser le composant/page `All-notes` représenté par l'ensemble de la figure 2 :

1. Ce composant demande à l'utilisateur de saisir dans un `input` le nom d'un cours. Lorsque l'on clique sur le bouton « soumettre », il exploite `MessageService` pour récupérer du backend la liste des couples (etudiant,note) correspondant dans la base de données.
2. Pour chaque couple (etudiant,note), on affiche le composant `Note` correspondant.
3. En dessous, on affiche le total des notes. Si l'utilisateur modifie les notes des étudiants, ce total doit être recalculé.
4. Enfin, si le backend a renvoyé une erreur, celle-ci est affichée dans la partie en rose indiquée en bas de la figure 2, sinon cette partie n'est pas affichée du tout.

Écrivez le contenu du template HTML ainsi que la classe TypeScript du composant `AllNotesComponent`.

Exercice 6 (1.5 points) — *requête SQL en node*

Écrivez l'équivalent Javascript de la fonction `getNotes` de l'exercice 1. Votre fonction renverra une `Promise` qui contiendra la liste (etudiant, note) des étudiants. Afin de requêter le serveur MySQL, on suppose que la fonction a accès à une variable « globale » `db` qui est déjà connectée avec la base de données (via l'instruction `const db = mysql2.createConnection(...)`).

Exercice 7 (1,5 points) — *Backend en Node*

Écrivez en javascript une fonction `notes(req,res)` équivalente au script de l'exercice 2 : elle récupère le nom du cours dans le body de la reqête et transmet les informations demandées au frontend via les fonctions `sendMessage` et `sendError` des TP (il est inutile de les réécrire pour l'examen).