Examen de première session

Durée : 1 heure 30

Documents autorisés : 1 feuille A4 recto-verso

Exercice 1 (3 points) — *Backend PHP – partie 1*

Une base de données contient une table `users` représentée dans la figure 1.

nom	prenom	age
Rabbit	Roger	30
...	...	...
Snow	John	28

FIGURE 1 – La table `users`.

On possède une instance `$pdo` de PDO qui est déjà connectée à la base de données et qui est une variable globale de notre programme. En utilisant celle-ci, écrivez en `PHP` une fonction `getUsers` qui prend en argument un nombre n et qui renvoie un tableau contenant les utilisateurs de la table `users` dont l'âge est inférieur ou égal à n. Chaque élément de ce tableau correspond à une personne et celle-ci est représentée sous la forme d'un tableau (au choix, associatif ou indexé) contenant juste son nom et son prénom.

Exercice 2 (3 points) — *Backend PHP – partie 2*

Écrivez un script `users.php` qui est appelé par un frontend via une méthode POST. Ce frontend transmet comme données une valeur n et votre script lui renvoie le tableau des personnes retourné par l'appel de la fonction `getUsers` de l'exercice précédent avec la valeur de n transmise par le frontend. Ce tableau est renvoyé au frontend au format JSON (vous pourrez avantageusement utiliser pour cela la fonction `json_encode`). Si le frontend a oublié de transmettre la valeur n, votre backend doit lui renvoyer le message « erreur valeur n ». Enfin, vous supposerez que vous avez à disposition un script `getUsers.php` qui contient le code de la fonction `getUsers` de l'exercice précédent.

Attention : ici, on ne veut que votre script `users.php`, n'écrivez pas le frontend ni son appel au backend.

Exercice 3 (3 points) — *Backend Node – partie 1*

Écrivez la fonction `getUsers` de l'exercice 1 en Node. Comme en TP, on veut que cette fonction renvoie une `Promise` qui contiendra le tableau des utilisateurs. Vous supposerez ici que votre programme contient une variable « globale » `db` qui est déjà connectée avec la base de données (via l'instruction `const db = mysql2.createConnection(...)`).

Exercice 4 (4 points) — *Backend Node – partie 2*

Comme dans l'exercice 2, on suppose que la fonction de l'exercice précédent est écrite et exportée dans un fichier `getUser.js`. On suppose également que les fonctions `sendMessage` et `sendError` des TP sont écrites et exportées dans le fichier `message.js`, comme en TP. On s'intéresse ici à réaliser une partie d'un backend Node/Express. Pour cela, on supposera que le frontend appelle votre backend via un `post` en utilisant la route `/users`. Il lui transmet la valeur n sous la forme d'un objet javascript `{n: valeur}` contenu dans le `body` de sa requête (comme vous l'avez vu en TP). Le backend lui renvoie alors la même réponse que dans l'exercice 2. Écrivez donc une fonction Javascript `users` qui s'occupe de traiter toute la requête du frontend (test de l'existence de n, envoi d'un message d'erreur s'il n'existe pas ou récupération de la liste des personnes et envoi de cette liste sinon). Indiquez également l'instruction Express qui permet d'appeler cette fonction quand le frontend envoie sa requête via un `post` sur la route `/users`.

Exercice 5 (7 points) — *Frontend en Angular*

On souhaite avoir un composant Angular permettant d'afficher la liste des personnes dans la base de données des exercices précédents dont l'âge est inférieur ou égal à une certaine valeur. Comme indiqué dans la figure 2, lorsque le browser charge ce composant, l'utilisateur n'ayant pas encore saisi de valeur pour l'*input* `Age`, le composant affiche le message « La liste est vide ». Lorsqu'il a saisi une valeur et cliqué sur le bouton « afficher », le composant demande au backend la liste des personnes et l'affiche, comme indiqué dans la figure 3. Vous supposerez que le backend est écrit en Node/Express et que la route pour y accéder est `http://127.0.0.1:3000/users` et vous ferez en sorte que votre requête au backend soit cohérente avec ce que vous avez écrit dans l'exercice précédent.
Écrivez le contenu du template HTML ainsi que la classe TypeScript de ce composant Angular (en ce qui concerne le .ts, vous n'indiquerez que le contenu de la classe. Il est donc inutile d'écrire les `import` ni le décorateur `@component`).

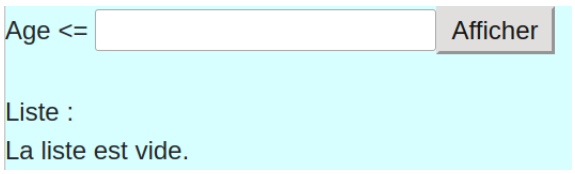

FIGURE 2 – Le composant Angular avant saisie de l'utilisateur.

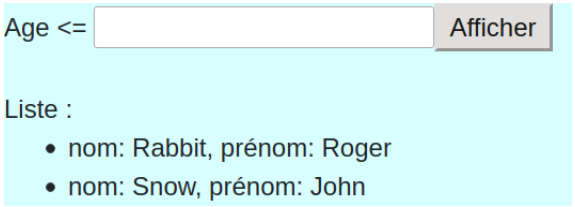

FIGURE 3 – Le composant Angular après avoir saisi un nombre et cliqué sur le bouton.