Examen de première session

Durée : 2 heures

Documents autorisés : La refcard du module

Exercice 1 (3 points) — *Backend en PHP*

La base de données de cet examen contient une table `users` représentée dans la figure 1.

nom	prenom	sexe
Banner	Bruce	H
Rabbit	Roger	H
Snow	John	H
Erhard	Emilia	F
Rabbit	Jessica	F
Romanoff	Natasha	F

FIGURE 1 – La table `users`.

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

1. un script `mysqlConnect.php` qui crée une instance `$pdo` de PDO, et qui la connecte à la base de données. Cette variable est une variable globale.
2. le script `helper.php` des TP, dans lequel on a supprimé la partie concernant l'authentification.

En exploitant ces scripts, écrivez dans le répertoire racine du backend un nouveau script `getUsers.php` qui est appelé par un frontend via une méthode **POST**. Ce dernier transmet à votre script une chaîne de caractères `sexe` au format `FormData`. Votre script réalise alors les opérations suivantes :

1. Il vérifie que la chaîne `sexe` a bien été transmise via le `POST` et que celle-ci est égale à « H » ou « F ». 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 (nom, prénom) des utilisateurs dont le genre correspond à la chaîne `sexe` (cf. l'interface `ListeData` de la figure 2). Par exemple, si `sexe` = H, la liste correspond aux 3 premiers éléments de la figure 1.
3. Il envoie un message de type `PhpData` (cf. figure 2) au frontend avec la liste calculée en 2.

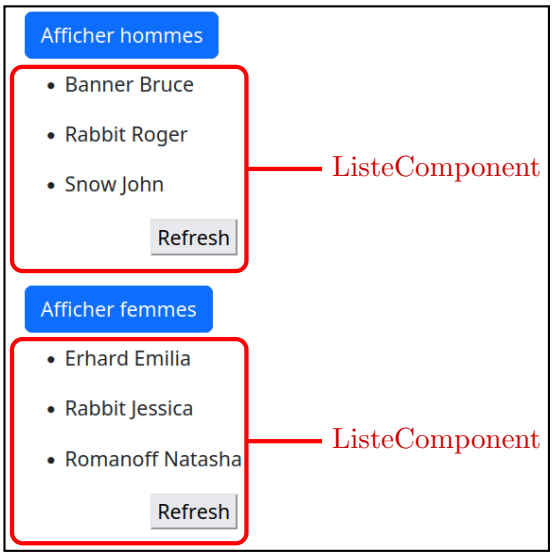
Ici, pour simplifier, aucune authentification n'est réalisée.

Exercice 2 (3 points) — *Frontend Angular – partie 1*

On souhaite développer une application Angular correspondant à la partie gauche de la figure 2. Celle-ci est constituée de deux composants :

— Le composant `ListeComponent` dont l'affichage correspond à l'intérieur des rectangles rouges.
— Le composant `PageComponent`, dont l'affichage correspond à la totalité de la partie gauche de la figure 2, et qui inclut deux instances de `ListeComponent`.

Le composant `ListeComponent` reçoit de son parent (`PageComponent`) le sexe des utilisateurs à afficher, sous la forme d'une chaîne de caractères « H » ou « F », ainsi qu'un tableau d'utilisateurs au format `ListeData` (cf. la figure 2). Il affiche ce tableau sous la forme d'une liste HTML. En dessous de la liste,




```
export interface ListeData {
   nom: string;
   prenom: string;
}


export interface PhpData {
   status: string;
   data: any;
}
```

FIGURE 2 – Application Angular et interfaces.

un bouton « Refresh » permet, quand on clique dessus, d'envoyer au parent un événement `refresh` contenant le sexe des utilisateurs affichés (« H » ou « F »).

Écrivez le contenu du template HTML ainsi que la classe TypeScript du composant `ListeComponent`. 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 3 (1 point) — *Frontend Angular – partie 2*

On a généré un service `MessageService` qui contient une méthode `sendMessage`. Celle-ci prend en argument une chaîne de caractères « H » ou « F » représentant le genre de l'ensemble des utilisateurs que l'on souhaite récupérer du backend. En utilisant une instance `http` de `HttpClient`, elle requête le script `http://127.0.0.1/getUsers.php` de l'exercice 1 et renvoie un `Observable` sur un `PhpData`. Écrivez cette méthode `sendMessage`.

Exercice 4 (5 points) — *Frontend Angular – partie 3*

Le composant `PageComponent` a deux boutons « Afficher hommes » et « Afficher femmes » qui changent l'état (affiché/caché) des `ListeComponent` qui sont juste en dessous d'eux : si une liste est affichée, cliquer sur le bouton au dessus la cache, et inversement. Par défaut, les deux instances de `ListeComponent` sont affichées. De plus, la page principale capture les événements `refresh` et, lorsqu'un tel événement arrive, elle recharge à partir du backend la liste concernée (et uniquement cette liste). Par exemple, s'il s'agit de la liste du bas de la figure 2, on ne recharge que la liste des femmes, pas celle des hommes.

Écrivez le contenu du template HTML ainsi que la classe TypeScript du composant `PageComponent`. Astuce : dans le TypeScript, écrivez deux méthodes `loadH()` et `loadF()` qui permettent de requêter le backend afin d'obtenir, respectivement, la liste des hommes et celle des femmes. N'oubliez pas, dans le template HTML, de passer aux `ListeComponent` les informations dont ces instances ont besoin et de capturer leur événement `refresh`.

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

Écrivez en javascript une fonction `getUsers` qui prend en argument une chaîne de caractères « H » ou « F ». Celle-ci requête alors la base de données MySQL pour récupérer la liste des utilisateurs de ce genre (au format `ListeData`) et, enfin, elle renvoie une `Promise` qui contiendra cette liste. Afin de requêter le serveur MySQL, on suppose que le programme contient une variable « globale » `db` qui est déjà connectée avec la base de données (via l'instruction `const db = mysql2.createConnection(...)`).

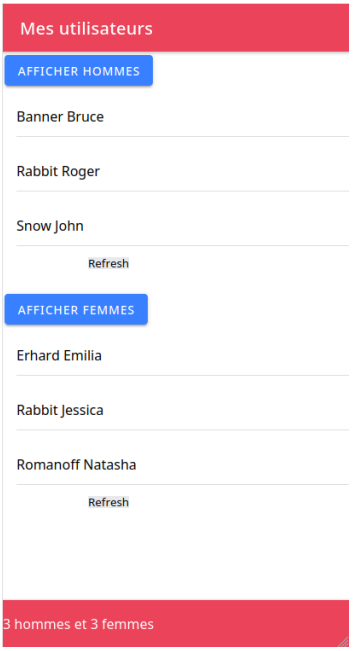
Exercice 6 (1,5 points) — *Backend en Node – partie 2*

On suppose que la fonction de l'exercice précédent est écrite et exportée dans un fichier `getUsers.js`. On suppose également que les fonctions `sendMessage` et `sendError` des TP sont écrites et exportées dans le fichier `message.js`. 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 `/myUsers`. Il lui transmet le genre des utilisateurs à afficher sous la forme d'un objet javascript `{sexe: valeur}` contenu dans le `body` de sa requête (comme vous l'avez vu en TP). Le backend lui renvoie alors la liste des utilisateurs ou un message d'erreur (via les fonctions `sendMessage` et `sendError`).
Écrivez une fonction Javascript `users` qui s'occupe de traiter toute la requête du frontend (test de l'existence de la propriété `sexe`, envoi d'un message d'erreur s'il n'existe pas ou récupération et envoi de la liste des utilisateurs sinon). Ici, on ne procédera à aucune authentification.

Exercice 7 (1 point) — *Backend Node – partie 3*

Indiquez l'instruction Express qui permet d'appeler la fonction de l'exercice 6 quand le frontend envoie sa requête via un `post` sur la route `/myUsers`.

Exercice 8 (4 point) — *Frontend Ionic*



On souhaite porter le frontend Angular des exercices précédents en Ionic. Pour cela, on utilise une page `PagePage` (qui correspond au composant Angular `PageComponent`) ainsi qu'un composant `ListeComponent` et un service `MessageService`. Leurs parties TypeScript sont exactement celles des exercices précédents et ne seront donc pas à réécrire.

Écrivez le template HTML du composant `ListeComponent` en Ionic.

Faites de même avec le template HTML de la page. Sur l'illustration à gauche, les parties en rouge sont toujours des hauts et bas de page. Notez que dans celle du bas, on indique le nombre d'hommes et de femmes recensées dans la base MySQL.