

Collège Ahuntsic
Département d'informatique

420-289-AH
Programmation web côté
serveur
hiver 2022

TP2

Développement d'une application météorologique

Énoncé

Satisfaite de votre première application Web, la firme MétéoWiki vous engage pour un nouveau contrat. Votre tâche principale consiste à migrer votre application initiale vers un projet Angular. Les besoins de la firme ont également changé, vous aurez donc à développer de nouvelles fonctionnalités.

Travail à faire

Étape 1: Migration de l'application [10 points]

La première étape de votre travail consiste à migrer les vues du TP1 (.hbs) vers l'application Angular se trouvant dans le dossier *client*. Assurez-vous de diviser vos vues en plusieurs composants afin de faciliter la maintenabilité de votre application.

Les onglets suivants devront être implémentés:

- **Conditions actuelles:** Onglet résumant les conditions actuelles d'une ou de plusieurs villes (voir le champ *current_condition*).
- **Prochaines heures:** Onglet résumant les conditions pour les prochaines. Chaque champ *weather* contient un champ *hourly*. Ce champ *hourly* indique les prévisions à une certaine heure. L'heure est disponible sous le champ *time* qui représente une certaine heure sous le format [militaire](#).

Les services nécessaire au fonctionnement de l'application Angular sont déjà présents. Vous pouvez donc directement utiliser les fonctionnalités disponibles dans *weather.service.ts*. Pour utiliser ce service, il vous suffit de l'injecter dans le constructeur de votre component.

```
constructor(private _weatherService: WeatherService) { }
```

Le getter *weatherSubject* est un [sujet](#) qui permet d'accéder aux dernières données acquises par le service. Ce sujet est utilisé pour éviter de devoir faire une requête HTTP à chaque fois qu'un component est créé. Assurez-vous que les données du service sont transmises entre vos vues et qu'une requête n'est pas envoyée par ce service à chaque changement d'onglet. **On veut donc conserver les villes recherchées entre chaque vue et éviter les requêtes inutiles ralentissant l'application.**

1.1 Création des composants (client)

Initialement, trois composants sont présents dans l'application et accessibles à partir d'une route définie dans le fichier `app.routing.module.ts`.

- <http://localhost:4200/weather/now> → Component pour l'onglet conditions actuelles
- <http://localhost:4200/weather/hourly> → Component pour l'onglet prochaines heures
- <http://localhost:4200/auth> → Component pour l'authentification (déjà fait pour vous)

Votre première tâche consiste à compléter les composants `weather-now` et `weather-hourly`.

1. Injecter le `weather.service` dans les composants.
2. S'abonner au `weatherSubject` et sauvegarder l'information météo dans une propriété (voir `TODO`). Ne pas oublier de se désabonner.
3. À partir de cette propriété et en vous basant sur vos `.hbs` du TP1, compléter le `.html` des deux composants.

Une fois cette étape complétée, vous devriez être en mesure d'afficher la météo de la ville par défaut dans chacun des composants.

1.2 Création d'un header (client)

Cette étape consiste à créer un component header pour votre client. Ce component doit contenir:

- Un logo provenant du dossier `assets`.
- Barre de navigation contenant les liens vers les onglets de l'application (**ne pas utiliser de `href` pour vos liens, utilisez `routerLink`**).
- Une barre de recherche.
- Un bouton pour lancer la recherche.

Afin de rendre la barre de recherche fonctionnelle, il vous suffit d'appeler la fonction `updateWeather` du `weather.service` en lui passant le contenu de la barre de recherche en paramètre. Vous devez donc:

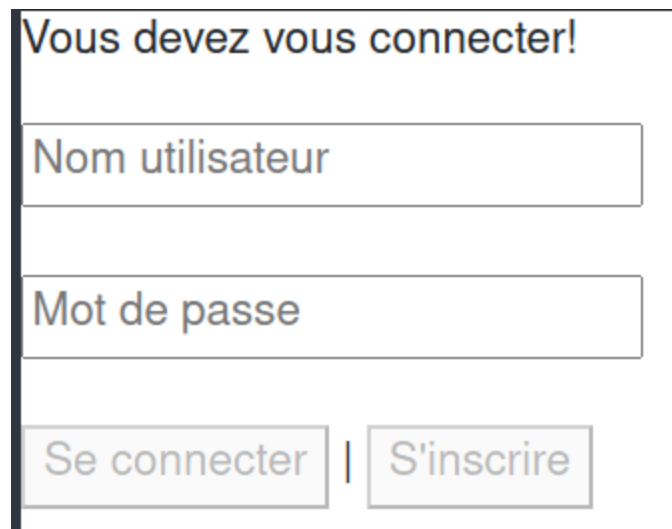
4. Générer un component header.
5. Compléter le `.html` du component header.
6. Créer un attribut recherche relié à la valeur de la barre de recherche.
7. Lier le bouton de recherche à une fonction du component. Dans cette fonction appeler la fonction `updateWeather` en lui passant l'attribut recherche en paramètre.

Ne pas oublier de s'abonner à l'observable retourné par la fonction sinon la requête ne sera jamais envoyée.

Une fois complété, vos composants devraient recevoir automatiquement les nouvelles informations de météo à la suite d'une recherche.

Étape 2: Gestion de l'authentification [10 points]

La deuxième étape consiste à ajouter l'authentification sur votre application. La portion à faire sur le client (Angular) est fournie. Le component *auth* est disponible à route */auth*. Vous pouvez ajuster le style du component en fonction du style de votre application.



Vous devez vous connecter!

Nom utilisateur

Mot de passe

Se connecter | S'inscrire

Les boutons "Se connecter" et "S'inscrire" permettent d'envoyer des requêtes au serveur (voir *auth.controller.ts* sur le serveur). Ces routes devront être complétées et une série d'étapes pour y arriver est fournie (voir les TODO). Vous devrez également compléter le service *mongodb.service.ts* (voir TODO) pour y arriver.

Assurez-vous que **le serveur empêche l'utilisation de la route */api/v1/weather*** si l'utilisateur n'est pas connecté. L'application Angular devrait également bloquer l'affichage des routes locales comme *weather/now* et rediriger l'utilisateur vers la page d'authentification. Pour cela, il vous suffit d'utiliser le guard fournis avec le TP dans la configuration des routes.

Fiez-vous aux différents TODO, ils pourront vous guider pour la résolution des deux étapes du TP. L'utilisation de l'extension *todo tree* est fortement recommandé.

2.1 Service MongoDB (serveur)

La première étape à suivre pour rendre l'authentification fonctionnelle consiste à compléter les deux fonctions du service MongoDB. Assurez-vous d'avoir MongoDB sur votre ordinateur. En utilisant la propriété *_collection* déjà initialiser, vous devez:

8. Compléter la fonction `getUserByUsername` (voir TODO).
9. Compléter la fonction `createUser` (voir TODO).

2.2 Routes authentification

Une fois le service MongoDB fonctionnel, vous pouvez compléter les deux routes dans le *auth.controller*. Ces routes sont utilisées par le service d'authentification du client.

10. Compléter la route POST <http://localhost:8000/api/v1/auth/signup> (voir TODO).
11. Compléter la route POST <http://localhost:8000/api/v1/auth/login> (voir TODO).
12. Tester vos routes avec Insomnia ou le component d'authentification.

2.3 Blocage des routes (client, serveur)

La dernière étape consiste à bloquer les routes du client et du serveur si le client n'est pas connecté. Vous devez donc:

13. Ajouter l'option *canActivate* à vos routes sur le client (*app.routing.module.ts*) en utilisant le guard *AuthGuard* (voir TODO).
14. Ajouter le *authHandler* à la route <http://localhost:8000/api/v1/weather> dans le *weather.controller* (voir TODO).
15. Tester l'authentification des routes clients et serveurs avant de se connecter.
16. Tester l'authentification des routes clients et serveurs une fois connecté.

Évaluation

Assurez-vous d'accorder de l'importance à la qualité de votre code avant d'effectuer votre remise:

- Ne pas laisser du code commenté qui ne sert à rien
- Ajouter des commentaires dans les sections complexes/critiques pour documenter votre code
- Ne pas laisser du code mort (code qui ne peut pas être exécuté comme du code à la suite d'un return)
- Éviter la duplication de code.

Remise

La remise de votre code se fera sur LÉA sous le format .zip avant le 3 avril 23h59. Assurez-vous de ne pas inclure les dossiers **node_modules** au .zip.

Vous pouvez faire le TP seul ou en équipe de 2.