

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](#).

Seuls les onglets **conditions actuelles** et **prochaines heures** sont obligatoires (vous n'êtes pas obligé de migrer **prochains jours** et **astronomie**).

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 composant.

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

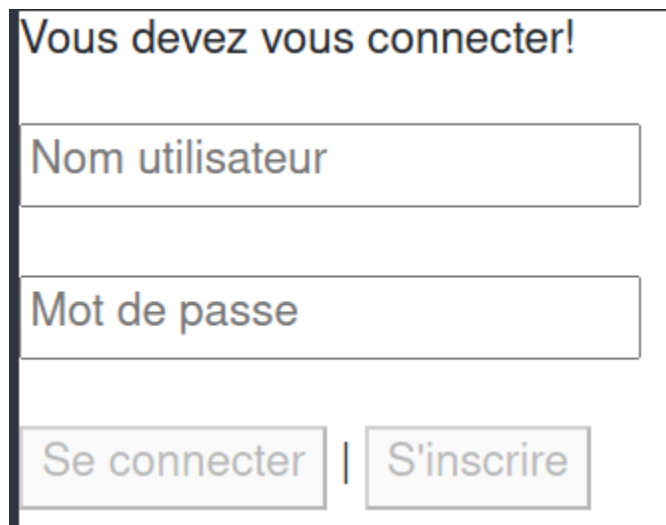
Le getter *data* est un sujet qui permet d'accéder à toutes les données acquises par le service. Assurez-vous que les données sont transmises entre vos vues et qu'une requête n'est pas envoyée par ce service à chaque changement d'onglet. **On veut donc conserver les villes recherchées entre chaque vue et éviter les requêtes inutiles ralentissant l'application.**

Attention, la route utilisée par ce service n'est pas complète. Vous devrez la compléter sur le serveur pour la rendre fonctionnelle. Vous pouvez retirer le paramètre *this.authHandler* au besoin pour retirer l'authentification de cette route pour vos tests.

```
// /api/v1/weather -> retourne la météo des locations
router.get('/', this.authHandler, async (req: Request, res: Response) => {
  // TODO utilisez req.locations et _weatherService.readWeathers
  // pour retourner la météo
});
```

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 composant *auth* est disponible à route */auth*. Vous pouvez ajuster le style du composant en fonction du style de votre application.



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 bloqué l’affichage des routes locales comme */weather* 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.

```
// TODO vous pouvez retirer temporairement le guard pour développer
{path: 'weather', component: AppComponent, canActivate: [AuthGuard]},
```

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.

Évaluation

Pour chacune des étapes, les critères suivants seront évalués:

- Qualité du code
 - Respect des règles du linter (*npm run lint*)
 - Documentation du code
- Fonctionnalité
 - Respect des exigences
 - Dépassement des attentes
- Appréciation globale
 - UI (interface utilisateur)
 - UX (expérience utilisateur)
 - Effort

Remise

La remise de votre code se fera sur LÉA sous le format .zip avant le 7 février 23h59. Assurez-vous de ne pas inclure les dossiers **node_modules** au .zip.