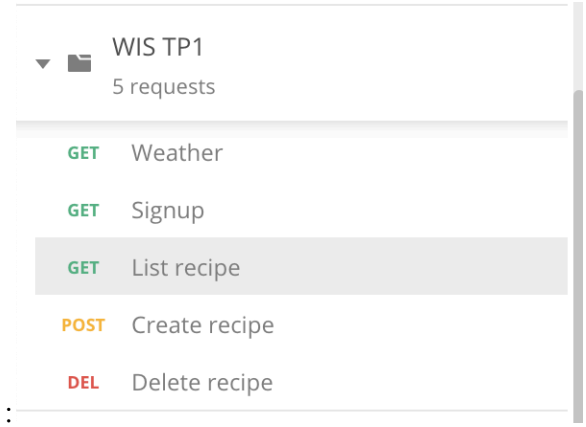
Exploiter une API

WIS 3 / Module Cloud et système web

L'objectif de ce TP est de créer une collection Postman regroupant des requêtes d'exemples vers des api publiques et privées.



Aperçu du résultat attendu

I. Exploiter une API publique

Récupérer des informations météorologiques sur OpenWeather

1. Télécharger l'application Postman : https://www.getpostman.com/
2. Créer une nouvelle Collection postman "WIS Cloud TP1"

Récupérer la météo avec l'api publique *openweather* :

3. Se rendre sur le site web de openweather : https://openweathermap.org/
4. Rechercher la documentation de l'API pour récupérer la météo actuelle à partir du nom d'une ville
5. Créer une nouvelle requête dans la collection postman et compléter les champs en suivant la documentation pour récupérer la météo de la ville de **Montpellier**

> *Comme la plupart des api publiques, une authentification est requise pour que le service définisse des quotas d'appels par utilisateurs (et rentabiliser son service sur les utilisateurs pros). Vous pouvez créer un compte gratuit (quelques minutes sont nécessaires avant que la clé soit activée) ou bien utiliser la clé d'api suivante :* ***3217255b4b8779d83bca28b97d9c88d9***

Exploiter d'autres api publiques

Découvrir des api publiques sur le site https://rapidapi.com/collection/list-of-free-apis
De la même manière que pour l'api open weather, créer quelques exemples de requêtes postman avec l'api de votre choix.

II. Exploiter une API privée

Nous utiliserons une api privée qui servirait à créer une application de recettes de cuisine.

Une API a été développée sur mesure pour ce tp. Visitez la documentation.

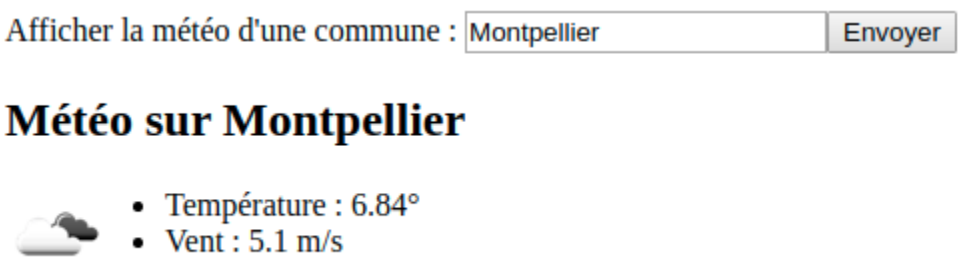
Endpoint: https://app-c3f0da02-0bb3-4b63-aaf9-964139bbab3d.cleverapps.io/api

En vous aidant de la documentation, créer les requêtes postman qui permettent les actions suivantes :

1. Créer un utilisateur
2. Créer une recette
3. Lister les recette existantes
4. Supprimer une recette

III. Exploiter une API dans un projet Angular

L'objectif est de créer une application de météo en utilisant l'API découverte dans la partie I. Nous utiliserons le framework Angular.



Initialisation du projet

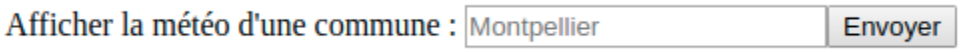
Créer un nouveau projet Angular en suivant ce guide : https://docs.google.com/document/d/1W4ibGTrS7WuJ07rV7t8CdtjW1_DO1jBqBSxo-u1LArU/edit?usp=sharing

Modules

Nous aurons besoin du module "FormsModule" pour le formulaire et de "HttpClientModule" pour les appels à l'api. Ajouter ces modules dans la partie "import" du fichier "app.module.ts".

Création du formulaire

Écrire dans le code html du composant "app.component.html" le code permettant l'affichage du formulaire suivant :



Relier le contenu du champ texte à une variable “city” (grâce à un ngModel) et le bouton “Envoyer” à une méthode “loadWeather” (événement click) dans le fichier typescript correspondant (app.component.ts).

Création du service

La bonne pratique est d’isoler les appels aux API dans un service. Créer un fichier service “openweather.service.ts” dans le dossier src/ qui centralisera les appels HTTP.

openweather.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
 providedIn: 'root'
})
export class OpenweatherService {

 BASE_URL = "http://api.openweathermap.org/data/2.5/"
 TOKEN = "9baeb2256032eb5d88358aac9577bae7";

 constructor(private client: HttpClient) { }

 getWeather(query: string) {
   return this.client.get(this.BASE_URL + 'weather?q=' + query + '&units=metric&APPID=' + this.TOKEN);
 }
}
```

Le code de l’appel à l’API vous est fourni. Vous pouvez vous en servir comme référence pour toutes les prochaines API à exploiter. L’usage est de déclarer les URL et Clés d’api en variable de classe, puis de créer une méthode par méthode d’api. Votre test d’api dans postman vous permet de connaître les paramètres à mettre en place. (Ici, ?q=, et &APPID=”)

Récupération des données et affichage

Dans le fichier weather.component.ts, déclarer une nouvelle variable de classe “weatherDatas” qui contiendra le retour de l’api. Dans la méthode loadWeather précédemment créée, ajouter l’appel au service :

weather.component.ts

```
….

weatherDatas: any;

loadWeather() {
 this.api.getWeather(this.city).subscribe((data) => {
   this.weatherDatas = data;
 });
}

...
```

Penser à injecter le service “api” OpenweatherService dans le constructor :

```
constructor(private api: OpenweatherService) { }
```

Vous pouvez maintenant utiliser le contenu de la variable “weatherDatas” dans votre template HTML pour former l’affichage. Utilisez la documentation de l’api ou votre test postman pour connaître le nom des propriétés disponibles. Exemple pour afficher la température : {{ weatherDatas.main.temp }}