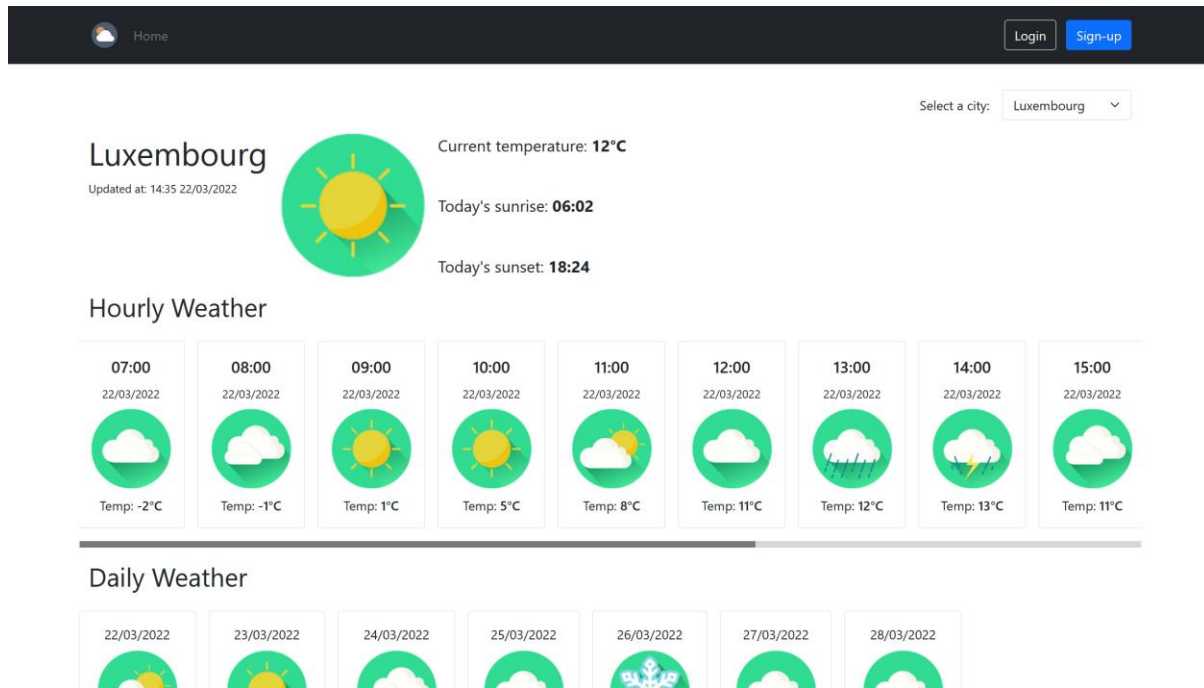


# TD6 (TYPE EXAM) – APPLICATION METEO

Le but de ce dernier TD est de réaliser une application météo basé sur une API fournie gratuitement par <https://open-meteo.com/>. Vous aurez à réaliser une page de vue générale de la météo d'une ville, ainsi qu'un petit formulaire d'inscription. Voici à quoi ressemble l'application :



Vous partez comme d'habitude d'une base en HTML/CSS/JS et vous devez construire à partir de celle-ci les fonctionnalités décrites dans ce document. Vous n'avez théoriquement pas besoin de toucher au HTML/CSS

## I/ LA SOURCE DE DONNEES ET CODE JS EXISTANT

### A) L'API OPEN METEO

Comme spécifié dans l'en-tête. Vous utiliserez l'API fournie par open-meteo. Vous pouvez accéder à la documentation de cette API ici : <https://open-meteo.com/en/docs>.

Regardez le détail si vous voulez. Mais essentiellement, vous aurez uniquement besoin d'utiliser la route suivante :

[https://api.open-meteo.com/v1/forecast?latitude=49.6076&longitude=6.0658&hourly=temperature\\_2m,weathercode&daily=weathercode,temperature\\_2m\\_max,temperature\\_2m\\_min,sunrise,sunset&current\\_weather=true&timezone=Europe%2FBerlin](https://api.open-meteo.com/v1/forecast?latitude=49.6076&longitude=6.0658&hourly=temperature_2m,weathercode&daily=weathercode,temperature_2m_max,temperature_2m_min,sunrise,sunset&current_weather=true&timezone=Europe%2FBerlin)

Cette route vous donnera toutes les informations dont vous avez besoin pour peupler votre page. **Vous devez juste remplacer la longitude et la latitude par les valeurs correspondant à la ville que vous voulez accéder.**

### B) LE FICHIER « TD6\_DATA.JS FOURNI »

Dû à la structure de données un peu particulière de cette API, vous avez à disposition un fichier « td6\_data.js ». Ce fichier vous met à disposition deux const clés pour vous aider avec ce mini projet :

```
1  const LOCATIONS = [  
2    { longitude: "13.4115", latitude: "52.5235", city: "Berlin" },  
3    { longitude: "2.3510", latitude: "48.8567", city: "Paris" },  
4    { longitude: "-0.1262", latitude: "51.5002", city: "London" },  
5    { longitude: "6.0658", latitude: "49.6076", city: "Luxembourg" },  
6    { longitude: "-3.7033", latitude: "40.4167", city: "Madrid" },  
7    { longitude: "16.3728", latitude: "48.2092", city: "Vienna" },  
8    { longitude: "4.3676", latitude: "50.8371", city: "Brussels" },  
9  ]
```

La constante « LOCATIONS » est un tableau de villes avec une longitude et latitude spécifiée. Elle vous servira à peupler votre sélecteur de ville par la suite, complété d'une longitude et latitude pour faire l'appel API correctement.

Par ailleurs, l'API vous retournera un code correspondant à la météo actuelle (ensoleillé, pluvieux, nuageux...) sous la propriété « weathercode » :

```
▼ current_weather:  
  windspeed:      8  
  temperature:    11.4  
  weathercode:    0  
  time:           "2022-03-22T11:00"  
  winddirection:  102
```

Ce code météo correspond à un standard pour désigner un état de météo. Il existe des tables pour associer ce chiffre avec un état « texte », mais afin de vous faciliter la conversion d'un code météo en image pour votre application, vous avez la méthode « getImageFromWMOCode »

```
70  const getImagePathFromNumber = (number) => {  
71    return `assets/weather-icons/Weather icons-${number}.png`;  
72  };  
73  
74  const getImageFromWMOCode = (code) => {  
75    switch (code) {  
76      case 0:  
77        return getImagePathFromNumber("02");  
78      case 1:  
79        return getImagePathFromNumber("05");  
80      case 2:  
81        return getImagePathFromNumber("03");  
82      case 3:  
83        return getImagePathFromNumber("04");  
84      case 45:  
85      case 48:  
86        return getImagePathFromNumber("09");  
87      case 51:  
88        return getImagePathFromNumber("13");  
89      case 53:  
90        return getImagePathFromNumber("19");  
91      case 56:  
92        return getImagePathFromNumber("25");  
93      case 63:  
94        return getImagePathFromNumber("30");  
95      case 65:  
96        return getImagePathFromNumber("31");  
97      case 71:  
98        return getImagePathFromNumber("50");  
99      case 80:  
100       return getImagePathFromNumber("60");  
101     }  
102   }
```

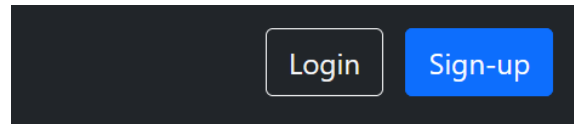
Vous aurez en retour le path « src » pour l'image correspondant au code passé en paramètre.

Utilisez ces méthodes et constantes pour faciliter votre développement de l'application.

## II/ TÂCHES À RÉALISER

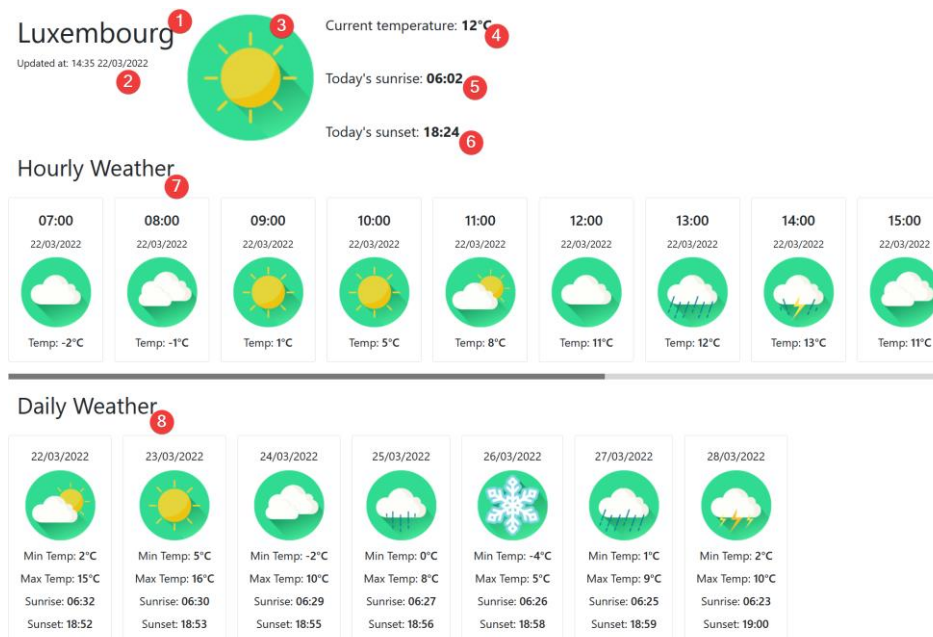
### A) LA PAGE "MÉTÉO"

- En haut à droite de la page, il y a un sélecteur de ville. Pour le moment, il y a 3-4 villes écrites en dur dedans avec aucune fonctionnalité. En partant du tableau « LOCATIONS » mis à disposition, peuplez cette select avec les villes fournies.



Select a city:

- Lorsqu'une ville est sélectionnée dans la dropdown, vous devez charger les informations météo basées sur ce pays et peupler la page basé sur le retour API. Pensez notamment à mettre à jour les informations dans les 8 sections suivantes :



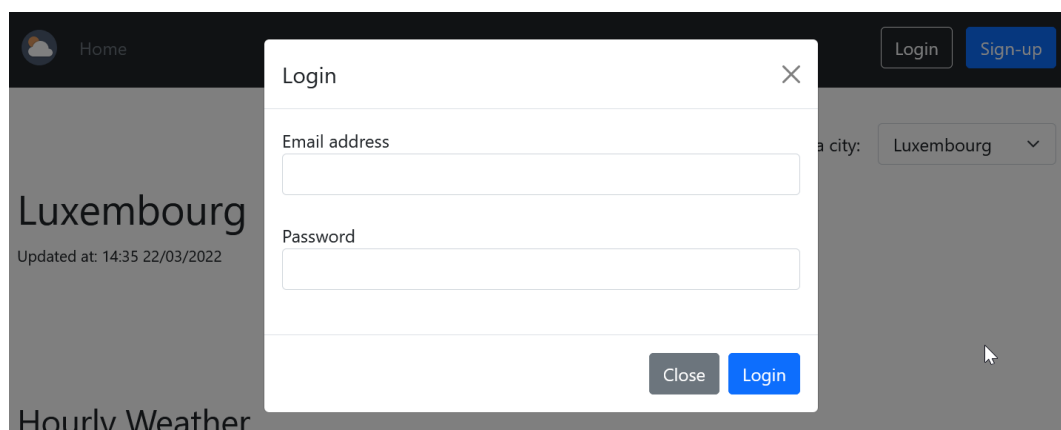
- Au chargement de la page en général, chargez les informations de la première ville dans la dropdown
- BONUS : Affichez un état intermédiaire sur la page lorsque les données sont en train de charger pour indiquer à l'utilisateur que les données sont en cours de chargement
- BONUS : Persistez la ville sélectionnée dans le [localStorage](#) du navigateur pour charger cette ville au chargement de la page

### B) LA MODALE DE LOGIN

Dans la page HTML, il y a un bout de code correspondant à une modale de login, cachée par défaut via son attribut style :

```
index.html X td6.css signup.html signup.css JS td6_data.js
TD06 > index.html > html
307 </main>
308 <div
309   class="custom-modal"
310   id="login-modal"
311   tabindex="-1"
312   style="display: none"
313 >
314   <div class="modal-dialog">
315     <div class="modal-content">
316       <form>
317         <div class="modal-header">
318           <h5 class="modal-title">Login</h5>
319           <button type="button" class="btn-close"></button>
320         </div>
321         <div class="modal-body">
322           <div class="mb-3">
323             <label for="email" class="form-label">
324               Email address
325             <input type="text" class="form-control" id="email" />
326             </label>
327           </div>
328           <div class="mb-3">
329             <label for="password" class="form-label">
330               Password
331             <input type="password" class="form-control" id="password" />
332             </label>
333           </div>
334         </div>
335         <div class="modal-footer">
336           <button
337             type="button"
338             class="btn btn-secondary"
339             data-bs-dismiss="modal"
340           >
341             Close
342           </button>
343           <button type="button" class="btn btn-primary">Login</button>
344         </div>
345       </form>
346     </div>
347   </div>
348 </div>
349 <script src="td06_data.js"></script>
```

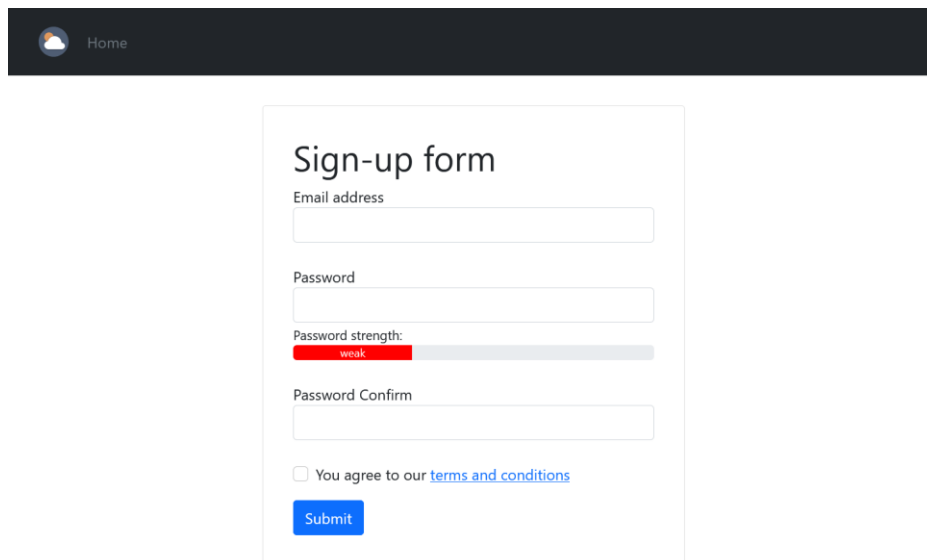
Lorsque cette modale est affichée dans la page, vous devrez voir ceci :



- Lorsque l'utilisateur clique sur le bouton « Login » en haut à droite de la page principale, affichez cette modale.
- La modale doit se fermer lorsque l'utilisateur clique sur le bouton « Close », la croix en haut à droite de la modale, ou bien sur el backdrop derrière la modale.
- Vous n'avez **PAS** besoin de gérer ce formulaire. Rien ne doit se passer lorsque l'on clique sur le bouton « Login »

### C) LA PAGE D'INSCRIPTION

Vous avez également une page d'inscription à faire avec un petit formulaire d'inscription avec quelques vérifications. La page se trouve sur « signup.html » :



Voici les tâches à faire sur cette page :

- Vérifiez si l'e-mail saisi est une adresse mail valide
- Le mot de passe doit faire au moins 8 caractères
- La confirmation de mot de passe doit correspondre au mot de passe
- La case termes et conditions doit être cochée
- Chaque champ erroné doit être indiqué clairement à côté du champ (il existe une balise HTML préexistante dans la page pour contenir les messages d'erreur)
- Empêchez l'envoi du formulaire si un champ n'est pas bon
- BONUS : Implémentez un fonctionnement pour la barre « password strength » juste en dessous du champ mot de passe. Suggestion d'implémentation : Une échelle allant jusqu'à 3 points, où il est possible de « gagner » des points lorsqu'on utilise des variétés de caractères différents. Par exemple, au moins une lettre minuscule donne 1 point, au moins une lettre majuscule donne 1 point, au moins 1 chiffre donne 1 point, et au moins 1 caractère spécial donne 1 point. Pensez à adapter le style avec la couleur de la barre, et le texte à afficher.

### III/ QUESTIONS THEORIQUES ET DEPOT DU TD

Veuillez déposer le TD en l'état à 12h le jour du TD. Déposez-le dans le formulaire suivant :

<https://forms.gle/wVS2zHvBNK9XRVp16>

Vous aurez également **2 questions théoriques** à remplir dans ce formulaire portant sur le JavaScript en général.