Développement d’un module de recherche, visualisation et réservation de bornes de recharge électrique

1. Contexte du projet

Vous travaillez sur le projet *Electricity Business*, une plateforme permettant à des particuliers d’accéder à des bornes de recharge de véhicules électriques. Vous devez réaliser une appli qui permet à un utilisateur :

- de rechercher des bornes autour d'une adresse,
- de visualiser les bornes sur une carte interactive ou sous forme de liste,
- de réserver une borne à une date et heure précises,
- de lister et supprimer ses réservations.

Le projet se fait entièrement côté client, sans backend.

2. Données utilisées

a. API Nominatim (géocodage d'adresse)

Documentation : https://nominatim.org/release-docs/develop/api/Search/

b. API Overpass (récupération de bornes)

- URL : `https://overpass-api.de/api/interpreter`
- documentation :
  - https://dev.overpass-api.de/overpass-doc/en/targets/formats.html#json
  - https://dev.overpass-api.de/overpass-doc/en/full_data/polygon.html#absolute_around
  - https://opendata.stackexchange.com/questions/12961/data-on-charging-station-for-electrical-cars
  - https://stackoverflow.com/questions/72192572/overpass-api-query-for-counting-amenity-of-specified-type-around-set-of-lat-lon

c. Classe Borne (et héritages)

Vous devez créer une hiérarchie de classes JavaScript comme suit :
`Borne`

`-> BornePublique` -> BornePrivée

- Une borne est publique si son `id` dans Overpass est pair, privée si impair.
- Les classes doivent exposer une méthode `toHTML()` personnalisée selon le type (`borne privée : id, lat, lon, proprietaire` , `Borne publique : id, lat, lon`).
- le propriétaire est un nom choisi au hasard parmi une liste

3. Fonctionnalités attendues (20pts)

Étape 1 – Recherche de bornes (7pts)

- Centrer une carte Leaflet sur votre géolocalisation.
- Créer un formulaire de recherche par adresse.
  - À la soumission utiliser Nominatim pour obtenir les coordonnées GPS (si vous n'y arrivez pas, voilà les coordonnées de 222 boulevard Gustave Flaubert, Clermont-ferrand : 45.75806298279684, 3.1270760116784317).
- Appeler Overpass pour récupérer les bornes dans un rayon de 5km (si vous n'y arrivez pas le document bornes.json contient les données des bornes à 5km des coordonnées du dessus).
  - Instancier chaque borne via `new BornePublique(...)` ou `new BornePrivee(...)` .
  - Afficher les bornes :
    - Sur la carte Leaflet via des marqueurs
    - Sous forme de liste HTML dans une autre section
- Ajouter un bouton "Basculer vue" pour afficher soit :
  - La carte interactive (Leaflet)
  - Une liste de bornes (DOM HTML)

Étape 2 – Réservation d’une borne (6pts)

- Ajouter une action “Réserver” pour chaque borne (clic sur le marqueur sur la carte et bouton sur la liste).
- Lors du clic :
  - Afficher un formulaire avec :
    - Date (date du jour par défaut)
    - Heure de début (format 24h)
    - Durée (entre 1 et 6 heures)
  - Valider les données :
    - Date future
    - Heure de début entre 6h et 22h

    - Durée conforme
- Créer une instance `Reservation` contenant :

```
{
  idBorne,
  typeBorne,
  date,
  heureDebut,
  duree
}
```

- Sauvegarder dans `localStorage`
- Afficher un message de confirmation

Étape 3 – Historique et suppression (5pts)

- Lire les réservations depuis `localStorage`
- Afficher l'historique dans un tableau :
    - ID de la borne, type, date, heure, durée
- Ajouter un bouton "Supprimer" par ligne
- Supprimer dynamiquement la ligne et mettre à jour le stockage

Étape 4 – Compte à rebours vers prochaine réservation (2pts)

- Afficher, en haut de page, le temps restant avant la prochaine réservation prévue.
    - Format : `hh:mm:ss`
    - Mise à jour chaque seconde
    - S'il n'y a aucune réservation future, afficher "Aucune réservation prévue"

4. Organisation du code

- Le code doit être structuré en modules
- Ajouter des commentaires clairs et cohérents
- Aucun framework, aucune bibliothèque js autre que Leaflet.js
- Le projet doit fonctionner hors ligne (en mode dégradé : affichage d'erreur si api indisponibles mais interface "lisible")
- L'interface doit être utilisable

5. Présentation orale (5pts soutenance, 5pts questions)

À la fin du développement, vous présenterez votre travail à l'oral. Cette présentation est notée et fait partie de votre évaluation. Elle comporte une partie présentation de 15 minutes maximum et question de 10 minutes maximum.

Conseils :

- Présentation du contexte
  - Expliquez brièvement l'objectif du projet *Electricity Business*.
  - Décrivez le besoin utilisateur que vous avez couvert.
- Structure du code et organisation
  - Présentez l'architecture des fichiers JavaScript, la hiérarchie des classes et leur logique d'instanciation.
  - Expliquez comment vous avez géré les appels API (Nominatim, Overpass) et la persistance des données avec `localStorage`.
- Choix techniques et difficultés rencontrées
  - Mentionnez une ou deux décisions techniques justifiées.
  - Partagez une difficulté que vous avez rencontrée et comment vous l'avez résolue.
- Démonstration fonctionnelle
  - Faites une démonstration complète d'un cas d'usage.
- Propositions d'amélioration
  - Présentez brièvement une ou deux idées d'évolutions.