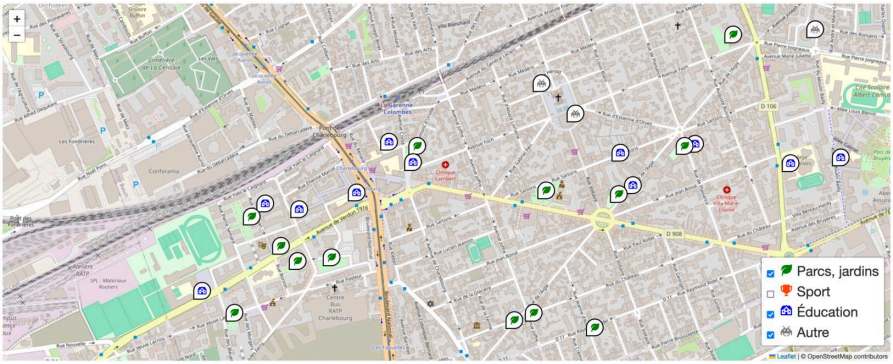
# 8

# Cas pratique : carte interactive

Dans ce projet, nous allons construire une carte interactive, qui va permettre de :

- mettre en place notre carte ;
- charger un fichier de données ;
- tracer des marqueurs personnalisés selon le type d'endroit désigné ;
- ajouter un filtrage pour ne voir que certaines catégories d'endroits.

***Figure 8.1 :*** *Le résultat final, avec les sports décochés*




## 8.1. Préparation des données

Nous allons pour ce projet utiliser les informations de la ville de La Garenne-Colombes, et positionner les points d'intérêt selon quatre catégories: les écoles (et collèges), les équipements sportifs, les parcs et jardins et une catégorie autre (pour la médiathèque, les marchés, etc.).

Nous allons créer un fichier JSON contenant les informations, et pour chacun d'entre eux nous allons les décrire ainsi :

```
[{
    "cat": "jardin", "lat": 48.9082, "lng": 2.2403,
    "nom": "square Jean-Boiselle"
  },
```

Les catégories seront nommées : jardin, école, sport, autre, et auront comme couleurs respectives le vert, le bleu, le rouge et le gris... de façon arbitraire.

## 8.2. Les marqueurs

Pour les marqueurs, nous allons utiliser une `div` que nous retravaillerons en CSS pour lui donner une forme arrondie avec une pointe, une bordure de couleur variable, ainsi qu'un pictogramme spécifique.

Pour ce point, on peut utiliser de nombreuses librairies disponibles sur Internet, dont le très fameux FontAwesome. Mais mon choix s'est porté sur Boxicons, entre autre pour son côté open-source (et accessoirement esthétique). Pour cela, il faudra ajouter à votre fichier HTML la ligne suivante.

```
<link
  href='//unpkg.com/boxicons@2.1.4/css/boxicons.min.css'
  rel='stylesheet'>
```

Partant de là, nous aurons une police permettant d'afficher des icônes via un code HTML ; il suffit de changer le nom de la classe (voir la documentation de Boxicons) pour changer l'allure de notre icône. La catégorie `autre` sera représentée avec un space invader, par manque d'inspiration... et un peu par nostalgie.

```
<i class="bx bxs-leaf"></i>
<i class="bx bxs-institution"></i>
<i class="bx bxs-trophy"></i>
<i class="bx bxs-invader"></i>
```

Nous allons commencer par créer une fonction `boxIcon` pour créer chaque marqueur. Notez que la pointe sera en bas à gauche.

```
function boxIcon(icon) {
  return L.divIcon({
    html: `<i class='bx bxs-${icon}'></i>`,
    iconSize: [30, 30],
    iconAnchor: [0, 30],
    popupAnchor: [15, -30],
```

```
    className: `boxiconsMarker`,
  });
}
```

Le paramètre envoyé sera le nom du pictogramme utilisé dans Boxicons. Deux classes CSS seront posées, une générique pour la forme des marqueurs, et de nouveau l'icône pour les déclinaisons de couleur.

```
.boxiconsMarker {
  border-radius: 50% 50% 50% 0; text-align: center;
  background: white; border: 2px solid black;
}
.boxiconsMarker i {
  width: 30px; height: 30px;
  font-size: 20px; line-height: 30px;
}
.bxs-leaf { border-color: green; color: green; }
.bxs-institution { border-color: blue; color: blue; }
.bxs-trophy { border-color: orangered; color: orangered; }
.bxs-invader { border-color: grey; color: grey; }
```

## 8.3. Charger les données, tracer les points

Charger les données depuis un JSON ne pose pas de réels problèmes.

```
fetch("data/92250-interactif.json")
  .then((response) => response.json())
  .then((data) => {
    let layers = traceMarkers(data);
    traceControl(layers);
  });
```

Une fois les données chargées, nous faisons appel à une première fonction `traceMarkers()` qui va tracer les marqueurs (!) et retourner la liste des layers (les quatre, un par catégorie). Nous enverrons alors les layers à un `traceControl()` pour tracer les contrôles (l'art de bien nommer ses fonctions...).

Il nous faut également une variable pour établir la correspondance entre le nom des catégories et le nom affecté dans Boxicons.

```
let catToBoxicon = {
  jardin: "leaf",       sport: "trophy",
  ecole: "institution", autre: "invader",
};
```