Coulibaly Cheick Ahmed
cheick7cca@gmail.com

Questions de révision

R1 : l'intervalle des valeurs pour les longitudes est [-180, 180] degrés.
R2 : l'intervalle des valeurs pour les latitudes est [-90,90] degrés.

R3 : Les coordonnées données sont **(-1.53388, 12.36566)**.
- La latitude est 12.36566 (car c'est toujours la deuxième valeur).
- La longitude est -1.53388 (car c'est toujours la première valeur)

R4 : La position du point **(-1.53388, 12.36566)** par rapport au point (0,0)

Le point (0,0) correspond à **l'intersection de l'équateur et du méridien de Greenwich**.

La latitude **12.36566** étant positive, le point est donc au Nord de (0,0).
La longitude **-1.53388** étant négative, le point est donc à l'Ouest de (0,0).

R5 : conversion en degrés minutes secondes
On applique la conversion :
Degrés : la partie entière.
Minutes : la partie décimale × 60.
Secondes : la partie décimale des minutes × 60.

Latitude : 12.36566
- 12° → partie entière.
- 0.36566 × 60 = 21.9396 → 21' (minutes).
- 0.9396 × 60 = 56.376 → 56.38" (secondes).

Résultat : 12.36566° = 12° 21' 56.38" N
N pour Nord

Longitude : -1.53388
1° → partie entière.
0.53388 × 60 = 32.0328 → 32' (minutes).
0.0328 × 60 = 1.97 → 1.97" (secondes).
Résultat : -1.53388 = 1° 32' 1.968" O
O pour Ouest.

Insertion de points dans Mongo

```
use uvbf
uvbf > db.createCollection("geoJSONtp2")
db.geoJSONtp2.insertMany([
... {name:"supermarché",location:{type:"Point",coordinates:[-1,-1]}},
... {name:"hôpital",location:{type:"Point",coordinates:[5,0]}},
... {name:"bureau",location:{type:"Point",coordinates:[0,-5]}},
... {name:"bar",location:{type:"Point",coordinates:[0.5,3]}}
... ]);

{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67c08f51a5260844272202d8'),
    '1': ObjectId('67c08f51a5260844272202d9'),
    '2': ObjectId('67c08f51a5260844272202da'),
    '3': ObjectId('67c08f51a5260844272202db')
```

```
  }
}
```

Rajout de l'index 2dsphere

```
db.geoJSONtp2.createIndex({ location: "2dsphere" })
```

L'index 2dsphere permet d'effectuer des requêtes géospatiales (calculs de distances, recherche de proximité, etc.). C'est le plus couramment utilisé pour stocker et interroger des coordonnées géographiques réelles (longitudes et latitudes).
Il permet les requêtes comme :
- trouver les points à proximité d'un lieu ($near, $geoWithin)
- vérifier si un point est dans une zone ($geoIntersects)

Les points par ordre croissant des distances par rapport à (0, 0)

```
db.geoJSONtp2.find({
... location:{
... $near:{$geometry:{type:"Point",coordinates:[0,0]}}
... }});
[
  {
    _id: ObjectId('67c08f51a5260844272202d8'),
    name: 'supermarché',
    location: { type: 'Point', coordinates: [ -1, -1 ] }
  },
  {
    _id: ObjectId('67c08f51a5260844272202db'),
    name: 'bar',
    location: { type: 'Point', coordinates: [ 0.5, 3 ] }
  },
  {
    _id: ObjectId('67c08f51a5260844272202d9'),
    name: 'hôpital',
    location: { type: 'Point', coordinates: [ 5, 0 ] }
  },
  {
    _id: ObjectId('67c08f51a5260844272202da'),
    name: 'bureau',
    location: { type: 'Point', coordinates: [ 0, -5 ] }
  }
]
```