420KBG – Laboratoire 02

Partir du projet suivant :

https://github.com/Nicolas-Chourot/API-Server---Solution---Labo-1

Idée générale

Les données d'API-Server sont stockées dans des fichiers JSON à travers la classe Repository. Cette dernière possède un méthode getAll() que vous devrez modifier afin que la querystring de la requête soit prise en compte pour filtrer et trier les données fournies dans la réponse.

? [nom du champ] = [filtre]	filtrer les données avec un critère de sélection sur un champ
? sort = [nom du champ], [desc]	trier les données sur un champ, option descendante

Exemple

```
/api/bookmarks                    voir tous les bookmarks
/api/bookmarks?sort=Name          voir tous les bookmarks triés ascendant par Name
/api/bookmarks?sort=Category      voir tous les bookmarks triés descendant par Category
/api/bookmarks/Id                 voir le bookmark Id
/api/bookmarks?Name=nom           voir le bookmark avec Name = nom
/api/bookmarks?Name=ab*           voir tous les bookmarks avec Name commençant par ab
/api/bookmarks?Name=*ab*          voir tous les bookmarks avec Name contenant ab
/api/bookmarks?Name=*ab           voir tous les bookmarks avec Name finissant par ab
/api/bookmarks?Category=sport     voir tous les bookmarks avec Category = sport
```

Note:

- Uniquement les noms des champs sont sensibles à la casse
- La querystring peut contenir plusieurs paramètres :

```
/api/bookmark?sort=Name,desc&Name=*e*&Category=news
```

Modalités de remise

Organisation :	Travail individuel
Date de remise :	**Lundi le 26 septembre 2022 à 23 h 59 (rencontre du 20 septembre dédié au laboratoire)**
Remise par Colnet	Une archive zip qui doit contenir : • Tous les fichiers sources • L'url de la version hébergée sur https://glitch.com/ dans le fichier html **`glitchLink.html`**

Annexe

Voici des méthodes qui vous seront fort utiles :

```
valueMatch(value, searchValue) {
    try {
        let exp = '^' + searchValue.toLowerCase();.replace(/\*/g, '.*') + '$';
        return new RegExp(exp).test(value.toString().toLowerCase());
    } catch (error) {
        console.log(error);
        return false;
    }
}

compareNum(x, y) {
    if (x === y) return 0;
    else if (x < y) return -1;
    return 1;
}

innerCompare(x, y) {
    if ((typeof x) === 'string')
        return x.localeCompare(y);
    else
        return this.compareNum(x, y);
}
```

Révision sur manipulation de collections d'objets ECMA 6

```
const items = [
    { name: 'Bike', price: 100 },
    { name: 'TV', price: 200 },
    { name: 'Album', price: 10 },
    { name: 'Book', price: 5 },
    { name: 'Phone', price: 500 },
    { name: 'Computer', price: 2000 },
    { name: 'Keyboard', price: 25 },
];
console.log(items);
console.log(items.length);
console.log(items[5].price);
// findIndex
console.log(items.findIndex(item => item.name === 'Computer'));
// destructuration
const [a, b, c] = items;
console.log(a, b, c);
// push
items.push({ name: 'Mouse', price: 60 });
console.log(items);
// pop
items.pop();
console.log("items.pop()");
console.log(items);
// shift
items.shift();
console.log(items);
// splice
items.splice(2, 2);
console.log(items);
// foreach
items.forEach(item => console.log(item.name + ' ' + item.price + '$'));
// sort
items.sort((a, b) => a.name > b.name ? 1 : -1);
console.log(items);
// copier et trier
const sortedPrices = [...items].sort((a, b) => a.price > b.price ? 1 : -1);
console.log(sortedPrices)
// filter
const filteredItems = items.filter(item => item.price <= 100);
console.log(filteredItems);
// map
const itemsNames = items.map(item => item.name);
console.log(itemsNames);
// find
const foundItem = items.find(item => item.name === 'Computer');
console.log(foundItem);
// some
const hasNotExpensive = items.some(item => item.price <= 100);
console.log(hasNotExpensive);
const hasFreeItems = items.some(item => item.price === 0);
console.log(hasFreeItems);
// every
const under2000 = items.every(item => item.price <= 2000);
console.log(under2000);
// reduce
const total = items.reduce((currentTotal, item) => { return item.price + currentTotal }, 0);
console.log(total);
// includes
const numbers = [1, 3, 5, 7, 9, 11];
const included = numbers.includes(6);
console.log(included)
```