

# self-service-machine-api - STEP 12

Dans cette étape nous allons :

- ordonner les résultats par ordre alphabétique
- améliorer la recherche

## Ordonner le résultat par ordre alphabétique

Commençons par ordonner le résultat de la route GET /api/products

```
...  
Product.findAll({ order: ["name"] })  
...
```

En ajoutant simplement `{ order: ["name"] }` nous faisons l'équivalent SQL de `ORDER BY name`.

Voilà le code complet :

```
...  
productsRouter.get("/", (req, res) => {  
  if (req.query.name) {  
    let limit = 3;  
    if (req.query.limit) {  
      limit = parseInt(req.query.limit);  
    }  
    return Product.findAndCountAll({  
      where: { name: { [Op.like]: `>${req.query.name}%` } },  
      order: ["name"],  
      limit: limit,  
    }).then((products) => {  
      const message = `Il y a ${products.count} produits qui correspondent au terme de la recherche`;  
      res.json(success(message, products));  
    });  
  }  
  Product.findAll({ order: ["name"] })  
    .then((products) => {  
      const message = "La liste des produits a bien été récupérée.";   
      res.json(success(message, products));  
    })  
    .catch((error) => {  
      const message =  
        "La liste des produits n'a pas pu être récupérée. Merci de réessayer dans quelques instants.";   
      res.status(500).json({ message, data: error });  
    });  
});  
...
```

Vous pouvez vérifier que le résultat est bien ordonné sur le nom du produit à la fois pour le résultat de la recherche mais également pour le résultat de tous les produits.

## Améliorer la recherche

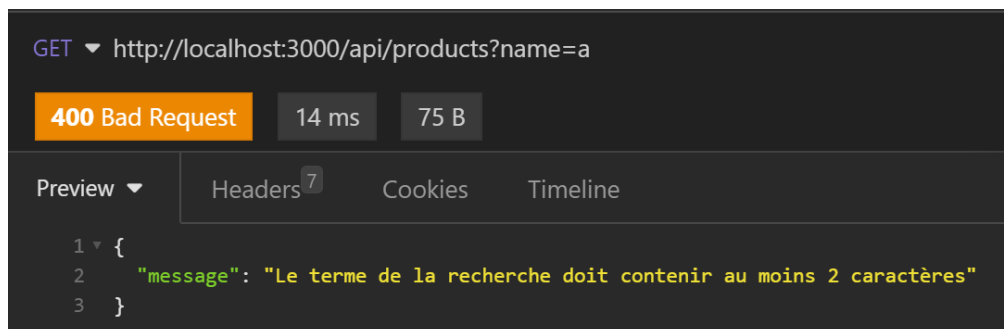
Si le critère de la recherche ne contient que 1 seul caractère, nous n'avons pas envie d'exécuter une recherche qui risque :

- d'être "gourmande" au niveau des performances

- de ne pas être bien utile pour notre consommateur

```
...
productsRouter.get("/", (req, res) => {
  if (req.query.name) {
    if (req.query.name.length < 2) {
      const message = `Le terme de la recherche doit contenir au moins 2 caractères`;
      return res.status(400).json({ message });
    }
    let limit = 3;
    if (req.query.limit) {
      limit = parseInt(req.query.limit);
    }
    return Product.findAndCountAll({
      where: { name: { [Op.like]: `%${req.query.name}%` } },
      order: ["name"],
      limit: limit,
    }).then((products) => {
      const message = `Il y a ${products.count} produits qui correspondent au terme de la recherche`;
      res.json(success(message, products));
    });
  }
  Product.findAll({ order: ["name"] })
    .then((products) => {
      const message = "La liste des produits a bien été récupérée.";
      res.json(success(message, products));
    })
    .catch((error) => {
      const message =
        "La liste des produits n'a pas pu être récupérée. Merci de réessayer dans quelques instants.";
      res.status(500).json({ message, data: error });
    });
});
...
```

Voilà ce que cela donne :



GET ▾ http://localhost:3000/api/products?name=a

**400 Bad Request** 14 ms 75 B

Preview ▾ Headers 7 Cookies Timeline

```
1 {
2   "message": "Le terme de la recherche doit contenir au moins 2 caractères"
3 }
```

Passons à l'étape n°13