Pagination, Tri et Recherche full-text dans une API

1. C'est quoi ?

Pagination : technique pour découper les résultats d'une requête en "pages" (lots limités d'éléments).\
Tri (sorting) : permet d'ordonner les résultats selon une ou plusieurs colonnes.\ **Recherche full-text** : permet de retrouver un élément à partir d'une portion de texte, même partielle.

2. Pourquoi on l'utilise ?

- Pour **éviter de surcharger** la mémoire et le réseau avec trop de résultats d'un coup.
- Pour offrir une **expérience utilisateur fluide** : affichage par lots dans une UI, ou scrolling infini.
- Pour **gagner en performance** sur de grosses bases (filtrage côté SQL).
- Pour permettre des recherches souples dans des champs texte.

3. Comment ça fonctionne ?

Pagination

On utilise souvent deux paramètres :

- `limit` : nombre maximum de lignes à retourner.
- `offset` : nombre de lignes à ignorer (skip).

Exemple SQL :

```sql
SELECT * FROM machines LIMIT 10 OFFSET 20;
```

Tri

Avec `ORDER BY` :

```sql
SELECT * FROM machines ORDER BY sector ASC, name DESC;
```

Recherche full-text simple (LIKE)

```sql
SELECT * FROM machines WHERE name LIKE '%to%';
```

4. Vocabulaire essentiel

Terme	Explication
`limit`	nombre d'éléments maximum à retourner
`offset`	nombre d'éléments à ignorer avant de commencer à retourner
`ORDER BY`	clause SQL qui permet de trier selon une ou plusieurs colonnes

Terme	Explication
`ASC / DESC`	ordre croissant ou décroissant
`LIKE`	comparaison sur du texte partiel (avec `%` comme wildcard)
`ilike`	version insensible à la casse (avec SQLAlchemy/FastAPI)

5. Syntaxe & Construction

En FastAPI :

```
from fastapi import Query

@app.get("/machines")
def listMachines(limit: int = Query(10), offset: int = Query(0), sortBy: str
= 'name', descending: bool = False):
    # construire la requête SQL avec LIMIT, OFFSET, ORDER BY
```

En cURL :

```
curl -X GET "http://127.0.0.1:8000/machines?
limit=10&offset=20&sortBy=sector&descending=true"
```

Pour la recherche texte :

```
curl -X GET "http://127.0.0.1:8000/machines?name=like:toto"
```

6. Représentation visuelle

```
Page 1 → limit=5 offset=0   → lignes 1 à 5
Page 2 → limit=5 offset=5   → lignes 6 à 10
Tri    → sortBy=year DESC   → les plus récents d'abord
```

✅Résultat attendu

Un endpoint FastAPI `/machines` (et `/logs`) capable de :

- **Limiter les résultats** (limit)
- **Avancer dans les pages** (offset)
- **Trier** (sortBy, order)
- **Chercher dans un champ texte** (LIKE)

Prochaine étape : Implémentation dans ta logique `dynamicRequest` avec une extension du parsing des `query_params` pour inclure :

- `limit`, `offset`

- `sortBy`, `descending`
- champs `LIKE` (ex: `name=like:toto`)