

Questions – Labo 3 – LOG430

Question 1 :

Les méthodes utilisant GET sont sûres et idempotentes, car elles ne modifient pas l'état du système et la répétition de la méthode ne change rien. Les méthodes utilisant POST sont non sûres et non idempotentes, car elles changent l'état du système (elles créent une ressource) et leur répétition crée plusieurs commandes. Enfin, les méthodes utilisant DELETE sont non sûres, car elles changent l'état du système, mais elles sont idempotentes, car leur répétition a le même effet final.

Question 2 :

Dans ce cas, on aurait pu utiliser soit *Simple Relationship Joins* ou *Joins to a Target with an ON Clause*. J'ai décidé d'utiliser la deuxième façon de faire en mentionnant explicitement la condition de liaison, mais cela n'était pas nécessaire. J'aurais pu le faire avec `.join(Stock.product)`.

```
def get_stock_for_all_products():
    """Get stock quantity for all products"""
    session = get_sqlalchemy_session()
    # TODO: ajoutez un join avec Product
    results = session.query(
        Stock.product_id,
        Stock.quantity,
        Product.name,
        Product.sku,
        Product.price,
    ).join(Product, Stock.product_id == Product.id).all()

    stock_data = []
    for row in results:
        stock_data.append({
            'Article': row.name,
            'Numéro SKU': row.sku,
            'Prix unitaire': row.price,
            'Unités en stock': int(row.quantity),
        })

    return stock_data
```

Question 3 :

Cette requête nous donne ce que le *body* demande donc l'id et la quantité du produit demandé.

The screenshot shows a GraphQL client interface with the following components:

- Method:** POST
- URL:** {{baseUrl}}/stocks/graphql-query
- Body Type:** GraphQL (selected), Auto Fetch (checked), Schema Fetched (status)
- Query:**

```
1 {
2   product(id: "1") {
3     id
4     quantity
5   }
6 }
7
```
- GraphQL Variables:** 1
- Status:** 200 OK, 24 ms, 224 B
- Response (JSON):**

```
1 {
2   "data": {
3     "product": {
4       "id": 1,
5       "quantity": 56
6     }
7   },
8   "errors": null
9 }
```

Question 4 :

Il faut regrouper les données des tables *Stocks* et *Products* en les liant avec *product_id* mais je n'ai pas été capable.

Question 5 :

Il faudrait voir tous les éléments appartenant aux deux tables donc *id*, *quantity*, *name*, *sku* et *price*

Question 6 :

Il partage les deux le même réseau (labo03-network) ce qui leur permet de communiquer ensemble.

```
1  services:
    ▶ Run Service
2  supplier_app:
3      build: .
4      environment:
5          - PYTHONUNBUFFERED=1
6      volumes:
7          - ./app
8      networks:
9          - labo03-network
10
11 networks:
12     labo03-network:
13         driver: bridge
14         external: true
15
```

```
1  services:
2      store_manager:
3          build: .
4          env_file:
5              - .env
6          environment:
7              - PYTHONUNBUFFERED=1
8          volumes:
9              - ../app
10         ports:
11             - "5000:5000"
12         networks:
13             - lab003-network
14         depends_on:
15             mysql:
16                 condition: service_healthy
17             redis:
18                 condition: service_healthy
19
```